About this project

This project focuses on a dataset that contains news articles along with their corresponding labels, indicating whether the articles are classified as real or fake news. The dataset includes several key attributes such as 'id,' which represents a unique identifier for each news article, 'title,' which provides the title of the article, 'author,' indicating the author of the news article, 'text,' which contains the actual content of the article, and 'label,' which assigns a value of 0 for real news and 1 for fake news.

The objective of this project is likely to build a predictive model or perform an analysis to distinguish between real and fake news articles. By utilizing the dataset and its attributes, various machine learning or natural language processing techniques can be applied to train a model that can accurately classify news articles as either real or fake.

The significance of this project lies in the importance of combating misinformation and disinformation in the media landscape. With the proliferation of digital platforms and the ease of sharing information, it has become crucial to develop effective methods to identify and address fake news. By leveraging the provided dataset, researchers or practitioners can gain insights into the characteristics and patterns associated with real and fake news, leading to the development of tools and strategies to mitigate the spread of false information.

Overall, this project is aimed at exploring and analyzing the provided dataset to develop a solution that can help in detecting and combating the presence of fake news, ultimately contributing to the promotion of accurate and reliable information in the media ecosystem.

Getting Started

About the dataset:

- id - unique identity for news article
- title - the title of the news article
- author - author of the news article
- text - the test of the article
- label - to mark the real vs fake article

0 = real news

1 = fake news

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

from matplotlib import rcParams
plt.rcParams['figure.figsize'] = [10, 10]
import seaborn as sns
sns.set_theme(style='darkgrid')
from wordcloud import WordCloud

import nltk
from nltk import sent_tokenize
from nltk.corpus import stopwords
stopwords = nltk.corpus.stopwords.words('english')
from nltk.tokenize import word_tokenize

import contractions
import re
import itertools
import datetime
import time
from collections import Counter
import string

import warnings
warnings.filterwarnings('ignore')


# remember to install some this dependencies
```

In [2]:
```
train = pd.read_csv('fake_news/train.csv')
test = pd.read_csv('fake_news/test.csv')

```

In [3]:
```
1 train.head()
```

Out[3]:

	id	title	author	text	label
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1

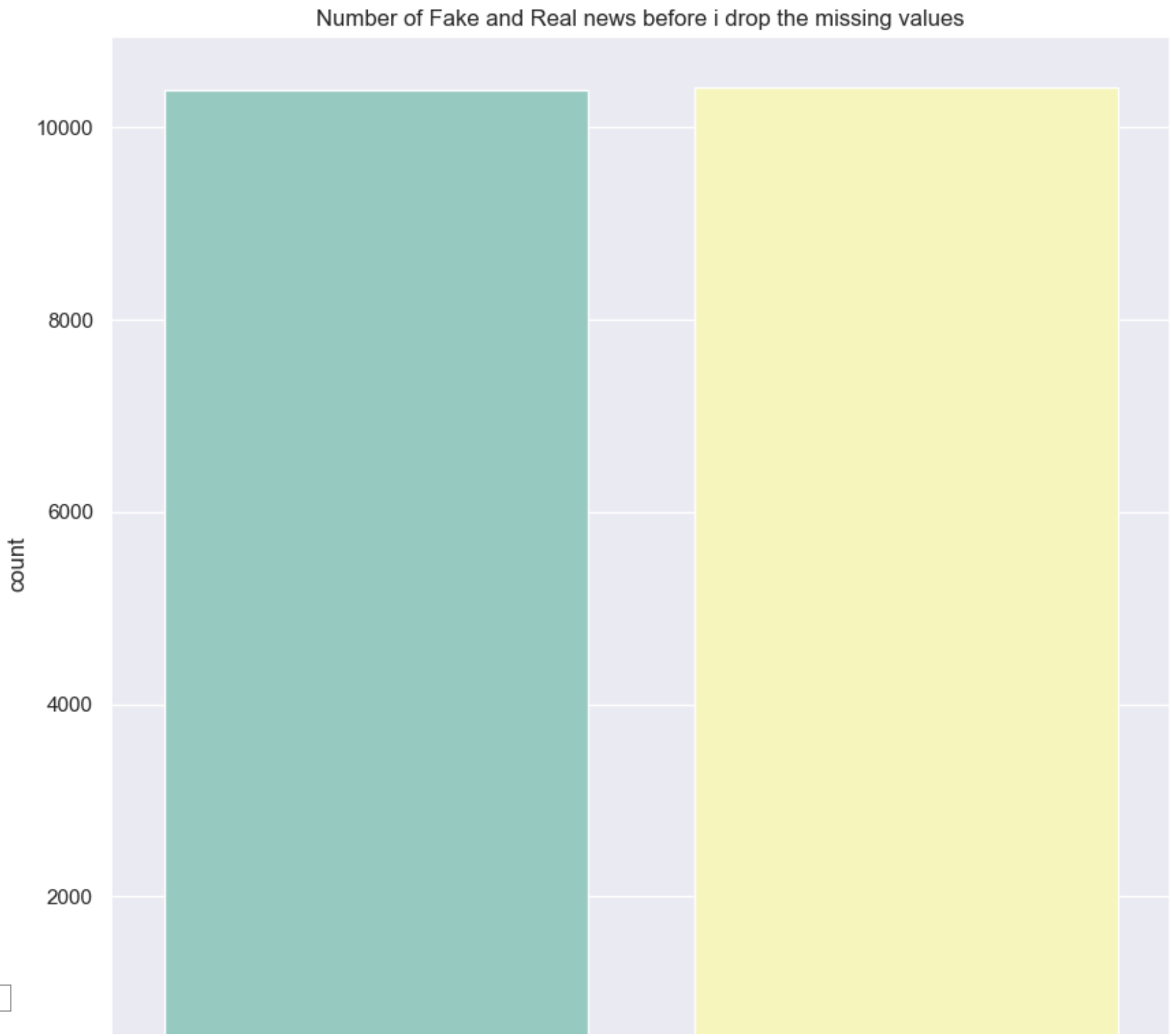
In [4]:
```
1 train.shape
```

Out[4]:
```
(20800, 5)
```

In [5]:
```
1 train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20800 entries, 0 to 20799
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      20800 non-null  int64 
 1   title   20242 non-null  object
 2   author  18843 non-null  object
 3   text    20761 non-null  object
 4   label   20800 non-null  int64 
dtypes: int64(2), object(3)
memory usage: 812.6+ KB
```

In [6]:
```
sns.countplot(x = 'label', data = train, palette = 'Set3')
plt.title('Number of Fake and Real news before i drop the missing values')
plt.show();
```



In [7]:
```
1 train.columns
```

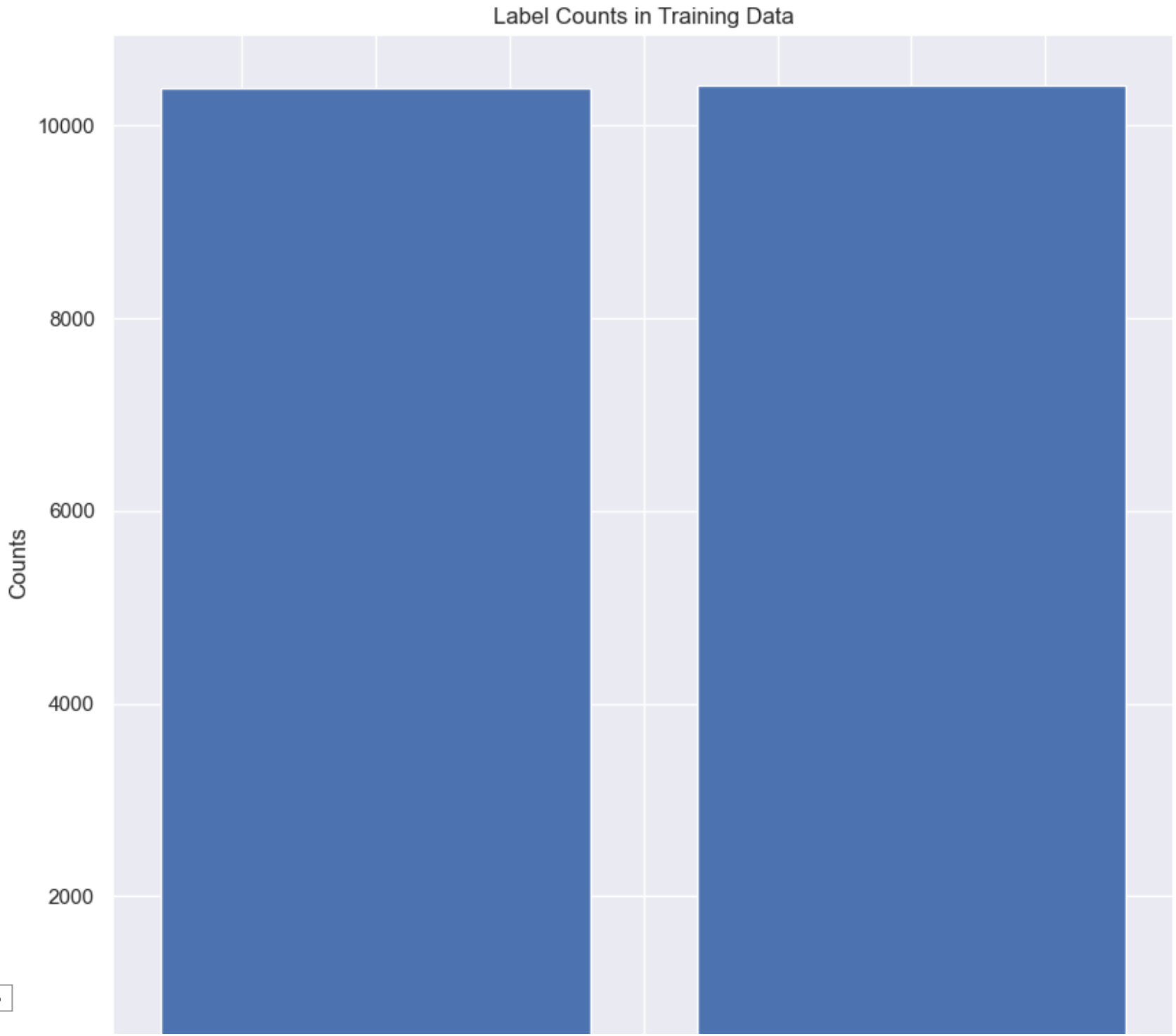
Out[7]:
```
Index(['id', 'title', 'author', 'text', 'label'], dtype='object')
```

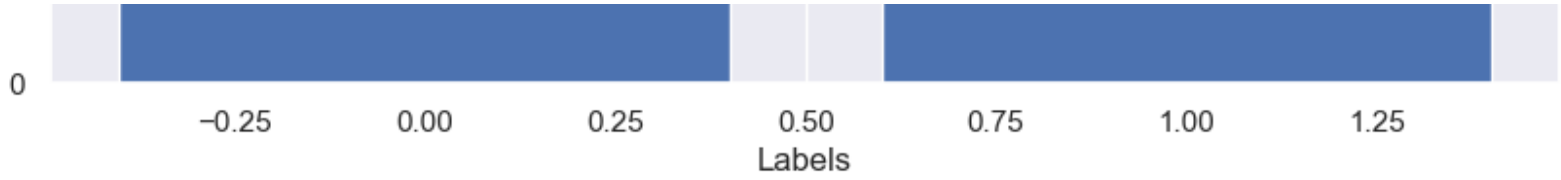
In [8]:
```
1 train['label'].value_counts()
```

Out[8]:
```
1    10413
0    10387
Name: label, dtype: int64
```

In [9]:

```
# Assuming 'train' is your DataFrame containing the 'label' column
label_counts = train['label'].value_counts()

# Plotting the bar chart
plt.bar(label_counts.index, label_counts.values)

# Adding labels and title
plt.xlabel('Labels')
plt.ylabel('Counts')
plt.title('Label Counts in Training Data')

# Display the chart
plt.show()
```





In [10]:
```
1 test.head()
```

Out[10]:

	id	title	author	text
0	20800	Specter of Trump Loosens Tongues, if Not Purse...	David Streitfeld	PALO ALTO, Calif. — After years of scorning...
1	20801	Russian warships ready to strike terrorists ne...	NaN	Russian warships ready to strike terrorists ne...
2	20802	#NoDAPL: Native American Leaders Vow to StayA...	Common Dreams	Videos #NoDAPL: Native American Leaders Vow to...
3	20803	Tim Tebow Will Attempt Another Comeback, This ...	Daniel Victor	If at first you don't succeed, try a different...
4	20804	Keiser Report: Meme Wars (E995)	Truth Broadcast Network	42 mins ago 1 Views 0 Comments 0 Likes 'For th...

In [11]:
```
1 test.shape
```

Out[11]:
```
(5200, 4)
```

In [12]:
```
1 test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5200 entries, 0 to 5199
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      5200 non-null   int64 
 1   title   5078 non-null   object
 2   author  4697 non-null   object
 3   text    5193 non-null   object
dtypes: int64(1), object(3)
memory usage: 162.6+ KB
```

In [13]:
```
1 test.isna().sum()
```

Out[13]:
```
id          0
title     122
author    503
text        7
dtype: int64
```

In [14]:
```
1 train.isna().sum()
```

Out[14]:
```
id           0
title      558
author    1957
text        39
label        0
dtype: int64
```

Dropping all instances which has atleast one column missing

In [15]:
```
1 train.dropna(axis = 0, how = 'any', inplace = True)
```

In [16]:
```
1 test = test.fillna(' ')
```

In [17]:
```
1 train.shape, test.shape
```

Out[17]:
```
((18285, 5), (5200, 4))
```

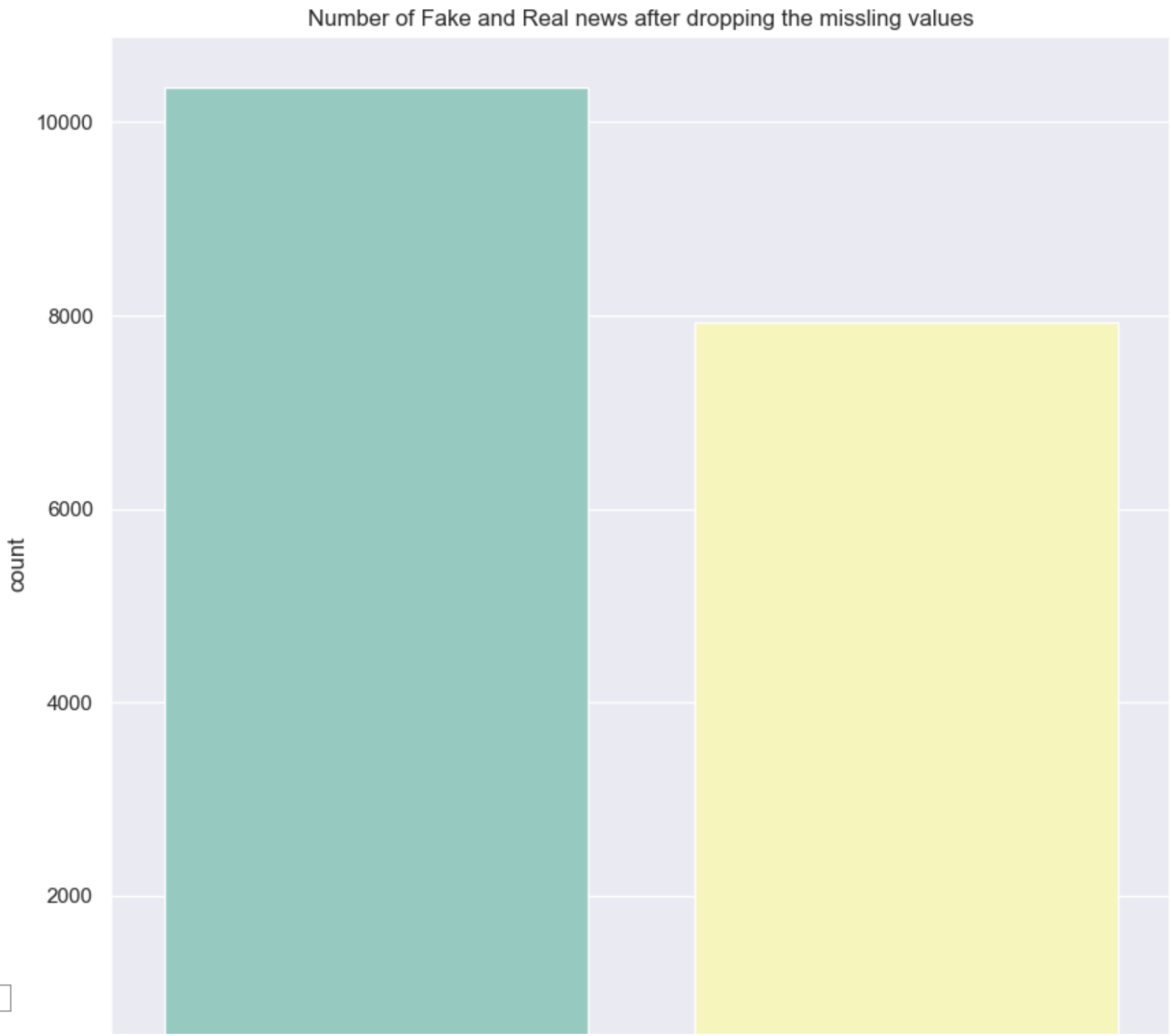
In [18]:
```
1 train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 18285 entries, 0 to 20799
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      18285 non-null  int64 
 1   title   18285 non-null  object
 2   author  18285 non-null  object
 3   text    18285 non-null  object
 4   label   18285 non-null  int64 
dtypes: int64(2), object(3)
memory usage: 857.1+ KB
```

In [19]:
```
1 test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5200 entries, 0 to 5199
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      5200 non-null   int64 
 1   title   5200 non-null   object
 2   author  5200 non-null   object
 3   text    5200 non-null   object
dtypes: int64(1), object(3)
memory usage: 162.6+ KB
```

In [20]:
```
sns.countplot(x ='label', data = train, palette = 'Set3')
plt.title('Number of Fake and Real news after dropping the missling values');
```



checking length of text

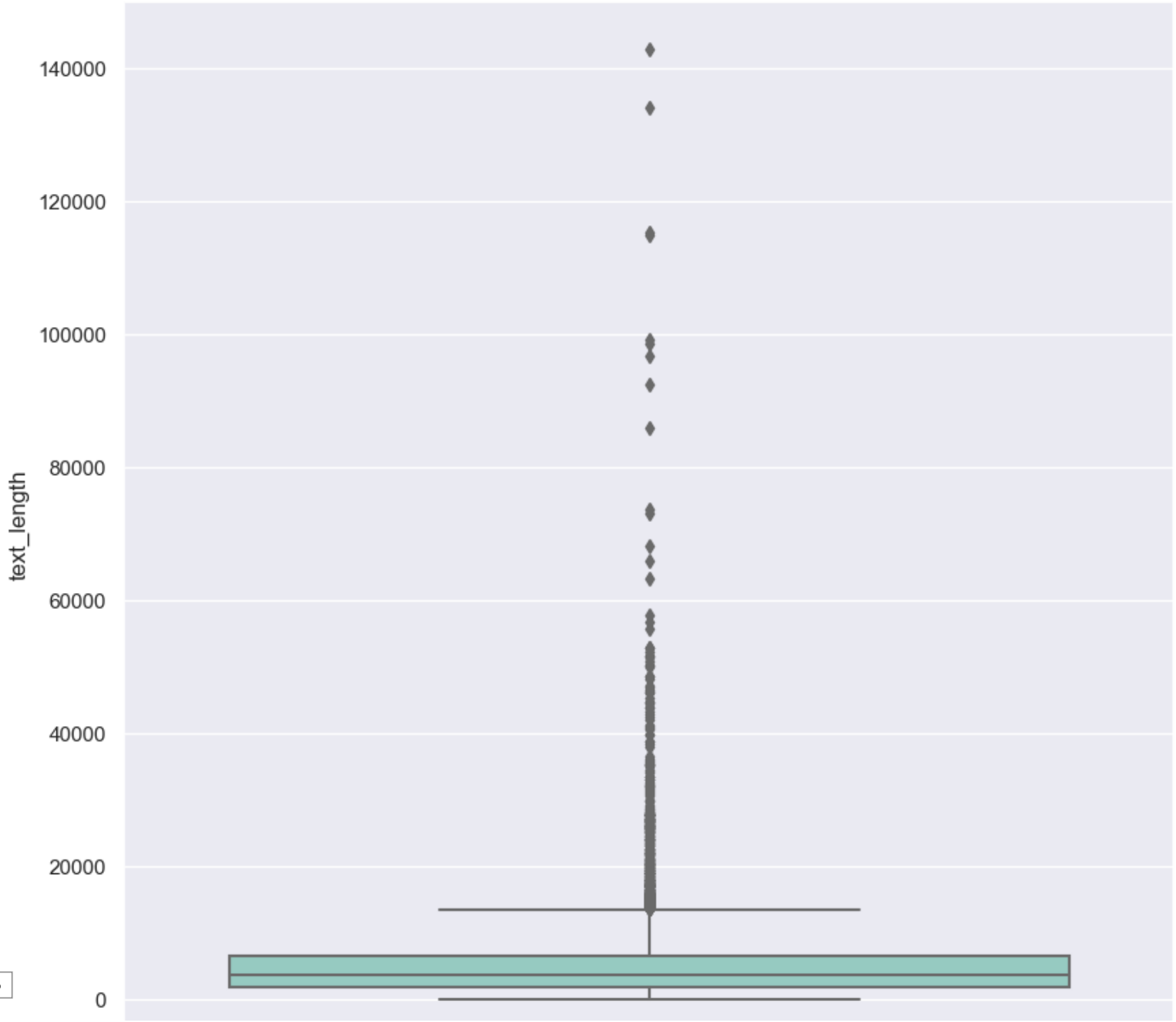
In [21]:
```
train['text_length'] = train['text'].apply(lambda x: len(x))
```

In [22]:
```
train.head()
```

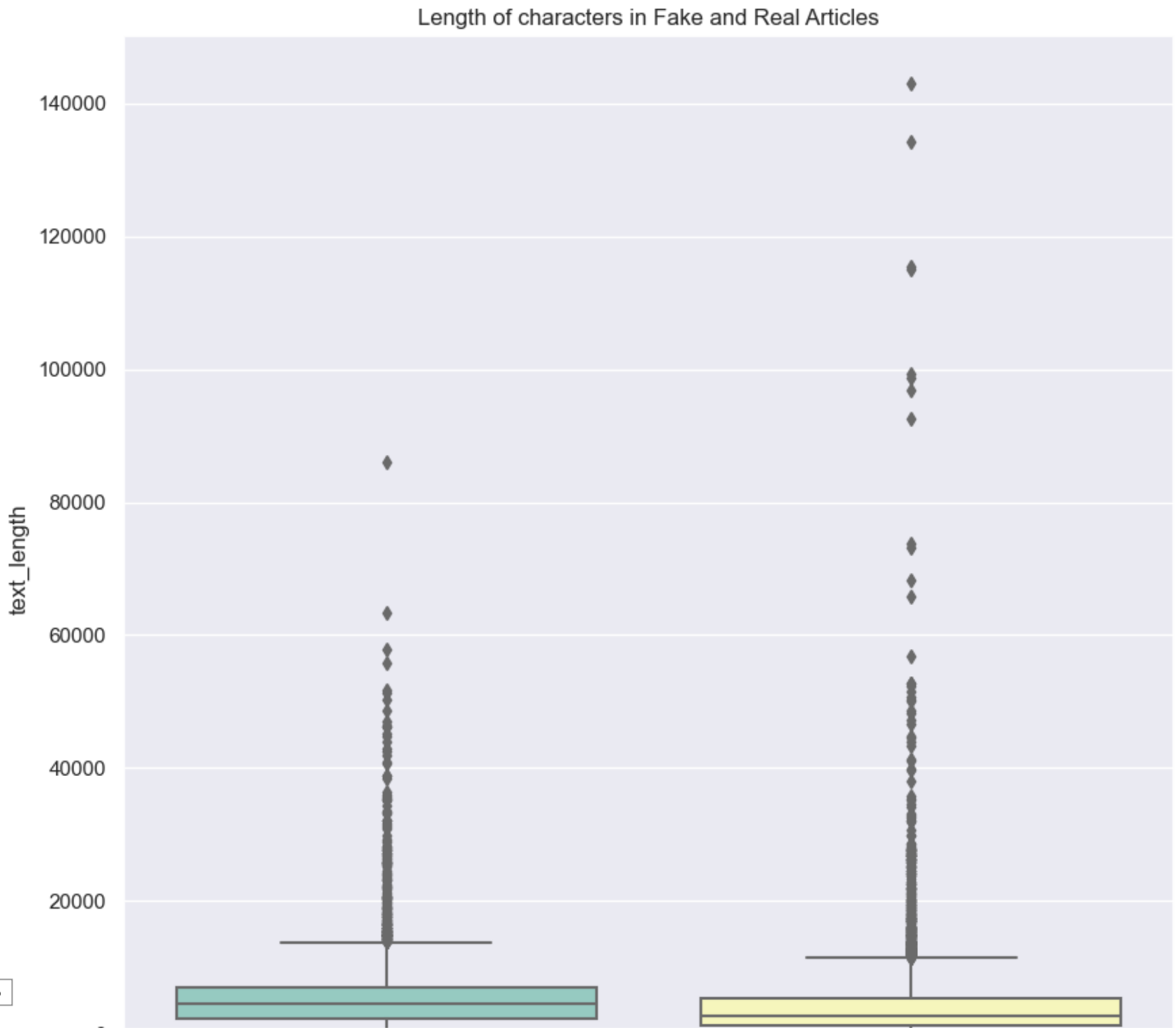
Out[22]:

	id	title	author	text	label	text_length
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	1	4930
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0	4160
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1	7692
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1	3237
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1	938

In [23]:
```
1 sns.boxplot(y = 'text_length', data = train, palette = 'Set3');
```



In [24]:
```
sns.boxplot(y = 'text_length', x = 'label', data = train, palette = 'Set3');
plt.title('Length of characters in Fake and Real Articles');
```



In [25]:
```
1 train['text_length'].describe()
```

Out[25]:
```
count     18285.000000
mean       4800.181843
std        5225.865069
min           1.000000
25%        1834.000000
50%        3694.000000
75%        6535.000000
max      142961.000000
Name: text_length, dtype: float64
```

checking how many rolls and columns of fake news

In [26]:
```
train[train['text_length'] == 1]
```

Out[26]:

	id	title	author	text	label	text_length
82	82	Huma’s Weiner Dogs Hillary	Steve Sailer		1	1
169	169	Mohamad Khweis: Another “Virginia Man” (Palest...	James Fulford		1	1
295	295	A Connecticut Reader Reports Record Voter Regi...	VDARE.com Reader		1	1
470	470	BULLETIN: There ARE Righteous Jews For Trump!;...	admin		1	1
592	592	Is your promising internet career over now Vin...	newsbiscuit editorial team		1	1
...	...	...	...	...	...	...
19857	19857	A Fifth Clinton Presidency? Hill, No!	Michelle Malkin		1	1
19929	19929	98% of public now ‘really looking forward’ to ...	NewsBiscuit		1	1
20242	20242	Radio Derb Transcript For October 21 Up: The M...	John Derbyshire		1	1
20264	20264	Pro-sovereignty Legislators Demand That Admini...	Brenda Walker		1	1
20513	20513	SAID IN SPANISH: A Mexican Governor Meddles In...	Allan Wall		1	1

73 rows × 6 columns

In [27]:
```
train['text'] = train['text'].str.strip()
```

In [28]:
```
train['text_length'] = train['text'].apply(lambda x: len(x))
```

In [29]:
```
train['text_length'].describe()
```

Out[29]:
```
count     18285.000000
mean       4799.886847
std        5225.922143
min           0.000000
25%        1834.000000
50%        3693.000000
75%        6535.000000
max      142961.000000
Name: text_length, dtype: float64
```

Pre-processing

i would rather replace the authors as blank and keep the text

In [30]:
```
df_train = pd.read_csv('fake_news/train.csv')
```

In [31]:
```
df_train = df_train.fillna(' ')
```

In [32]:
```
df_train.isna().sum()
```

Out[32]:
```
id        0
title     0
author    0
text      0
label     0
dtype: int64
```

The code df_train['text'] = df_train['text'].str.strip() is performing an operation on a column called 'text' in a DataFrame called df_train.

The .str.strip() method is being applied to each element in the 'text' column. This method removes leading and trailing whitespace from each string in the column.

By assigning the result back to the 'text' column (df_train['text'] = ...), the original column is modified, and all the strings in the 'text' column are stripped of any leading or trailing whitespace.

In [33]:
```
df_train['text'] = df_train['text'].str.strip()
```

In [34]:
```
1 df_train.head()
```

Out[34]:

	id	title	author	text	label
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1

In [35]:
```
1 df_train['text_length'] = df_train['text'].apply(lambda x: len(x))
```

In [36]:
```
1 df_train.head()
```

Out[36]:

	id	title	author	text	label	text_length
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	1	4930
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0	4160
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1	7692
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1	3237
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1	938

In [37]:
```
1 len(df_train[df_train['text_length']== 0])
```

Out[37]: 116

In [38]:
```
1 df_train = df_train[df_train['text_length'] > 0]
```

In [39]:
```
1 df_train.head()
```

Out[39]:

	id	title	author	text	label	text_length
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	1	4930
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0	4160
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1	7692
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1	3237
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1	938

In [40]:
```
1 df_train.shape
```

Out[40]:
```
(20684, 6)
```

In [41]:
```
1 df_train['text_length'].describe()
```

Out[41]:
```
count     20684.000000
mean       4569.360472
std        5132.617915
min           1.000000
25%        1641.750000
50%        3384.500000
75%        6288.250000
max      142961.000000
Name: text_length, dtype: float64
```

In [42]:
```
df_train[df_train['text_length'] == 10]
```

Out[42]:

	id	title	author	text	label	text_length
5048	5048		Anonymous	Brilliant!	1	10
8920	8920		Anonymous	Brilliant!	1	10
10784	10784		Jonathan white	are u dumb	1	10
12993	12993	2009 FLASHBACK: “What If” Remixed	AlexAnsary	11/08/2016	1	10
15582	15582		Anonymous	Brilliant!	1	10
16929	16929		Anonymous	Brilliant!	1	10
18006	18006		william ketley	she is fit	1	10

Thes look like comments

In [43]:
```
len(df_train['author'].unique())
```

Out[43]: 4196

In [44]:
```
real_news_authors = set(list(df_train[df_train['label'] == 0]['author'].unique()))
fake_news_authors = set(list(df_train[df_train['label'] == 1]['author'].unique()))
```

In [45]:
```
1 real_news_authors
```

Out[45]:
```
{'Azam Ahmed, Paco Nunez and Alan Blinder',
 'Jonathan Wolfe',
 'Eric Lipton, Noah Weiland and Steve Eder',
 'Simon Romero and Andrew Jacobs',
 'David E. Sanger and Nicole Perlroth',
 'Michael S. Schmidt, Mark Mazzetti and Matt Apuzzo',
 'Barry Meier and Susanne Craig',
 'John Markoff and Matthew Rosenberg',
 'Tanya Mohn',
 'Abby Goodnough and Jonathan Martin',
 'Nick Wingfield, Mike Isaac and Katie Benner',
 'Mitch Smith, Rukmini Callimachi and Richard Pérez-Peña',
 'Jad Mouawad',
 'Christopher Mele and Monica Davey',
 'Alan Feuer',
 'John Koblin and Michael M. Grynbaum',
 'Steve Knopper',
 'Zach Schonbrun',
 'Bob Price',
```

In [46]:
```
1 len(real_news_authors)
```

Out[46]:
```
2226
```

In [47]:
```
1 fake_news_authors
```

Out[47]:
```
{'Svetlana Arkhangelskaya, special to RBTH',
 'Beanie',
 'James M. Dorsey',
 'Sue Penn (@SuePennonTwitte)',
 'James',
 'Jamieson',
 'freeinaz',
 'Financial Times',
 'Contributed Daily News Bin Staff',
 'The Rundown Live',
 'Gordon Duff, Senior Editor',
 'Christy Lee Parker',
 'Patrick J. Buchanan',
 'Kit Daniels',
 'Ness Nomadison',
 'Whitney Webb',
 'RIA Novosti, Pavel Gaikov',
 'E. A. Costa',
 ':ohemad: (UID 73271507)',
```

In [48]:
```
1 len(fake_news_authors)
```

Out[48]:
```
1976
```

In [49]:
```
1 overlapped_authors = real_news_authors.intersection(fake_news_authors)
```

In [50]:
```
1 overlapped_authors
```

Out[50]:
```
{' ', 'AFP', 'Ann Coulter', 'Pam Key', 'Pamela Geller', 'Reuters'}
```

In [51]:
```
1 len(real_news_authors), len(fake_news_authors), len(overlapped_authors)
```

Out[51]:
```
(2226, 1976, 6)
```

In [52]:
```
1 df_train.head()
```

Out[52]:

	id	title	author	text	label	text_length
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	1	4930
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0	4160
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1	7692
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1	3237
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1	938

In [53]:
```
1 train_df = df_train.copy()
```

In [54]:
```
1 train_df.head()
```

Out[54]:

	id	title	author	text	label	text_length
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	1	4930
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0	4160
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1	7692
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1	3237
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1	938

Text Columns Pre-processing

1. Remove special characters
2. Expand contractions
3. convert to lower-case
4. word tokenize
5. Remove stopwords

Remove special characters: This step aims to remove any special characters, such as punctuation marks and symbols, from the text. It helps clean the text data by eliminating unnecessary noise and focusing on the essential words.

Expand contractions: Contractions are shortened versions of words or phrases. Expanding contractions involves converting them back to their original forms. For example, "can't" would be expanded to "cannot." This step ensures consistency and improves the understanding of the text.

Convert to lower-case: Converting the text to lowercase standardizes it by making all the letters in the text lowercase. This step ensures that words with different letter cases are treated as the same entity during analysis. For instance, "Hello" and "hello" would both become "hello."

Word tokenize: Tokenization involves splitting the text into individual words or tokens. This process breaks down the text into meaningful units, allowing for further analysis. Each word becomes a separate element in a list. For example, the sentence "Hello, how are you?" would be tokenized into ["Hello", ",", "how", "are", "you", "?"].

Remove stopwords: Stopwords are commonly used words that do not carry significant meaning in a particular context. Examples include "the," "and," "is," etc. Removing stopwords helps to eliminate noise and focus on the more important words in the text.

By applying these steps together, the code performs a basic text pre-processing pipeline that cleans, standardizes, and prepares the text data for subsequent analysis or natural language processing tasks.

In [55]:
```
def preprocess_text(x):
    cleaned_text = re.sub(r'[^a-zA-Z\d\s\']+', '', x)
    word_list = []
    for each_word in cleaned_text.split(' '):
        try:
            word_list.append(contractions.fix(each_word).lower())
        except:
            print(x)
        return " ".join(word_list)

```

In [56]:
```
# rearranging order of preprocessing
text_cols = ['text', 'title', 'author']
```

In [57]:
```
%%time
for col in text_cols:
    print('Processing column : {}'.format(col))
    df_train[col] = df_train[col].apply(lambda x: preprocess_text(x))
    test[col] = test[col].apply(lambda x: preprocess_text(x))
```

```
Processing column : text
Processing column : title
Processing column : author
Wall time: 9.94 s
```

Typesetting math: 0%

%%time: This is a Jupyter Notebook magic command that measures and prints the execution time of the code cell. It allows you to assess the time taken by the code to run.

for col in text_cols:: This line indicates that there is a list or iterable called text_cols containing the names of columns to be processed. The loop iterates over each element in text_cols, assigning it to the variable col in each iteration.

print('Processing column: {}'.format(col)): This line prints a message indicating the name of the column currently being processed. The curly braces {} are a placeholder for the value of col, which is inserted into the string using the format() method.

train_df[col] = train_df[col].apply(lambda x: preprocess_text(x)): This line applies a preprocessing function called preprocess_text() to each element in the column col of the DataFrame train_df. The result of the preprocessing is assigned back to the same column col in the DataFrame train_df. The apply() function is used to apply the preprocess_text() function to each element in the column.

test[col] = test[col].apply(lambda x: preprocess_text(x)): Similarly to the previous line, this line applies the preprocess_text() function to each element in the column col of the DataFrame test. The result is assigned back to the same column col in the DataFrame test.

Overall, the code snippet performs text preprocessing on the specified columns (text_cols) in two DataFrames (train_df and test). The preprocess_text() function is assumed to be defined elsewhere and is applied to each element in the specified columns using the apply() function.

Word_tokenize

In [58]:
```
%%time
for col in text_cols:
    print('Processing column : {}'.format(col))
    df_train[col] = df_train[col].apply(word_tokenize)
    test[col] = test[col].apply(word_tokenize)
```

```
Processing column : text
Processing column : title
Processing column : author
Wall time: 11.3 s
```

Tokenization is a natural language processing (NLP) technique that breaks down a text or a sequence of characters into smaller units called tokens. These tokens can be words, subwords, or even characters, depending on the level of granularity required for the task at hand.

Tokenization is an essential preprocessing step in many NLP tasks because it helps convert raw text into a format that is easier to analyze and process. Here are a few reasons why tokenization is important:

1. Text analysis: Tokenization allows you to analyze text at a more granular level, treating each token as a separate unit. This can be useful for tasks such as sentiment analysis, part-of-speech tagging, named entity recognition, and more.
2. Vocabulary creation: Tokenization helps in building the vocabulary for a language model or a machine learning algorithm. Each unique token becomes a distinct entry in the vocabulary, which enables the model to learn and make predictions based on those tokens.
3. Text normalization: Tokenization can also be a part of the text normalization process, where the text is transformed to a standardized format. For example, converting all characters to lowercase or removing punctuation marks can be done at the token level.
4. Text representation: Tokenization plays a crucial role in representing text as numerical data that can be processed by machine learning algorithms. Once the tokens are created, they can be further encoded using techniques like one-hot encoding, word embeddings (such as Word2Vec or GloVe), or contextual embeddings (such as BERT or GPT) to capture semantic information.

Overall, tokenization is a fundamental step in NLP that breaks down text into smaller meaningful units, enabling various analysis, modeling, and processing tasks in the field of natural language processing.

The given code appears to be using the `%%time` magic command, which is typically used in Jupyter notebooks to measure the execution time of the code cell.

The code is then using a loop to iterate over each column specified in the `text_cols` variable. For each column, it prints a message indicating the column being processed.

Within the loop, the code applies the `word_tokenize` function to each element in the corresponding column of the `train_df` DataFrame using the `apply` method. The `word_tokenize` function is likely a tokenization function that splits a string of text into individual words or tokens. The resulting tokenized text is then assigned back to the same column in `train_df` .

The same tokenization process is repeated for the corresponding columns in the `test` DataFrame, and the tokenized text is assigned back to those columns as well.

Overall, this code is tokenizing the text data in specific columns of the `train_df` and `test` DataFrames, and it provides a progress message for each column being processed. The `%%time` magic command is used to measure the execution time of the entire code cell.

In [59]:
```
1 df_train.head()
```

Out[59]:

	id	title	author	text	label	text_length
0	0	[house]	[darrell]	[house]	1	4930
1	1	[flynn]	[daniel]	[ever]	0	4160
2	2	[why]	[consortiumnewscom]	[why]	1	7692
3	3	[15]	[jessica]	[videos]	1	3237
4	4	[iranian]	[howard]	[print]	1	938

Stopwords

In [60]:
```
1 # import os
2 # os._exit(00)
```

In [61]:
```
1 %%time
2 for col in text_cols:
3     print('Processing column : {}'.format(col))
4     df_train[col] = df_train[col].apply(lambda x: [each_word for each_word in x if each_word not in stopwo
5     test[col] = test[col].apply(lambda x: [each_word for each_word in x if each_word not in stopwords])
6
```

```
Processing column : text
Processing column : title
Processing column : author
Wall time: 531 ms
```

In [62]:
```
df_train.head()
```

Out[62]:

	id	title	author	text	label	text_length
0	0	[house]	[darrell]	[house]	1	4930
1	1	[flynn]	[daniel]	[ever]	0	4160
2	2	[]	[consortiumnewscom]	[]	1	7692
3	3	[15]	[jessica]	[videos]	1	3237
4	4	[iranian]	[howard]	[print]	1	938

In [63]:
```
train_df.isna().sum()
```

Out[63]:
```
id             0
title          0
author         0
text           0
label          0
text_length    0
dtype: int64
```

Wordcloud

In [64]:
```
df_train['text_joined'] = df_train['text'].apply(lambda x: " ".join(x))
test['text_joined'] = test['text'].apply(lambda x: " ".join(x))
```

In [65]:
```
df_train.head()
```

Out[65]:

	id	title	author	text	label	text_length	text_joined
0	0	[house]	[darrell]	[house]	1	4930	house
1	1	[flynn]	[daniel]	[ever]	0	4160	ever
2	2	[]	[consortiumnewscom]	[]	1	7692	
3	3	[15]	[jessica]	[videos]	1	3237	videos
4	4	[iranian]	[howard]	[print]	1	938	print

In [66]:
```
all_text_real = " ".join(df_train[df_train['label'] == 0]['text_joined'])
all_text_fake = " ".join(df_train[df_train['label'] == 1]['text_joined'])
```

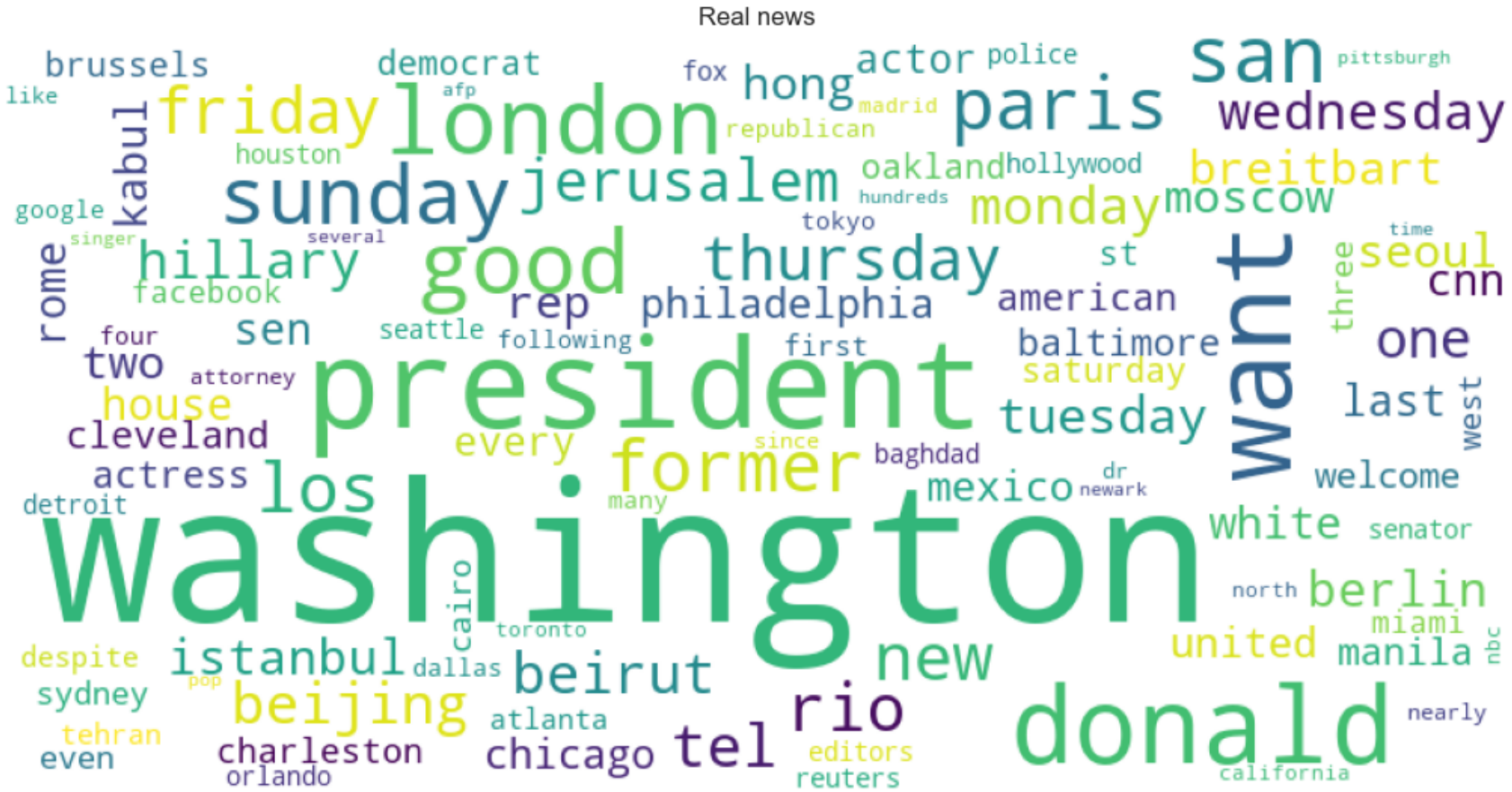
In [67]:
```
all_text_fake
```

Out[67]:
```
"house  videos print ever  clinton yes  fbi email massachusetts email humiliated country  open    0 0 dueli
ng sounds samantha  click 1   17 oregon clinton  22  email bob 0  share  whether license  trump   breaking
zero   chapo     politics 21st editors    un google crushing serena geert business comments  tweet think fl
uoridation lt comments donate  guest sunday comments carey teacher  fromthefront uk  1 20 death wed  home
9   clinton  new pokemon remember  print october  ask la get changing us  dr  fbi former follow   ya posted
posted share waking  getty ok report wed altright    wed  us october pginas  zach email nevada 79 home unit
ed trumps november las tweet lazy share  90 november 2016 another trump comments report  citizen 161 doctor
s  2016 0 jewsnews sonoma tweet posted  print nuclear geoengineering gambia  eu 4  education carey wed trum
p chinese print posted posted     breaking clintons breaking refugee tuesday  interwebs share share 2016 em
ail october voters posted 1212 images october references note jane reunin hallowe'en   ed country media jew
snews street videos  support blue markets   police contaminated scandal   cyber  us tourist zero email alab
ama national 0 source  usa 2700yearold  4 2016 considering insurance 0  nteb hillary donald  syrian  0 arou
nd  october 8  field  trump report guys  africa julian november man  go 45 ludicrous leave region 0 first
november posted comments email share vladimir america videos  home  dr november email turkish  comments bil
l home russia email comments  syria videos statins society 19 taming corbett next  posted posted 0 illegal
email  52     breakdown  fifth  trump share thank francis  share 330   judgment wars   0  waking   0   emai
l  physorg  russia hillary click  tuesday 30 lukas email      carol  october trump email  56   brother satu
rated lisa f thu print  0 donald email hillary colorado 0 home le  link memes schools  pinterest video   l
eave  october sweden hemp editor 's scientists     wed  next nteb published wikileaks russian   posted vide
```

In [68]:
```
1 all_text_real
```

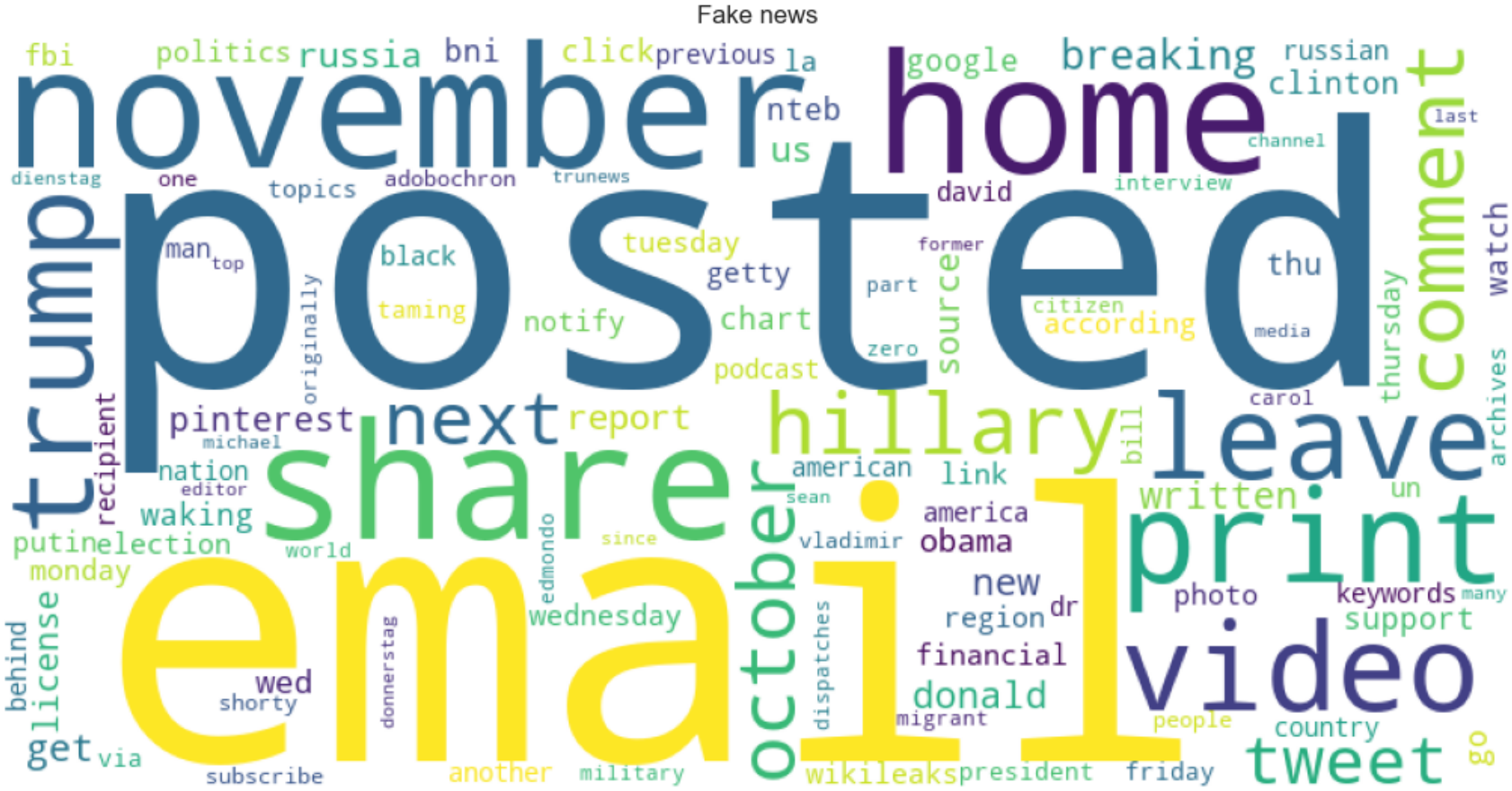
Out[68]:
```
'ever  paris donald  organizing  guillermo  wednesday screenwriter sunday orders jerusalem andrea hillary
london midland  vorokhobino rep     executives san   editors  three  washington washington  new hangzhou  w
ashington sunday want ap  mary denver   iquitos  president   three  donald attorney  texas  barbara  snapch
at  dallas ottawa marital washington hong greenbelt kabul news house  aston democrats rio illegal four desp
ite normani washington washington washington    tokyo birmingham police sunday   cairo normally  beirut stu
ck steve   donald miami   president cleveland  former los controversy washington liverpool  washington rodn
ey  harrison washington well washington megyn   cnn  sean halloween hong palm pew thursday national valenti
na washington london  senator cairo   donald  chattanooga eleven reuters reverend  lumberton     berlin rio
president davos dueling doubling russia washington president   washington companies jerusalem  donald washi
ngton taking louisville good sixty senate sydney matt  amid   sam according hong president bestselling ches
ter washington policing want  arianna differentiating  projected washington sunday  washington london storr
s  luxembourg stephen laughter wednesday san oakland washington   remnants barack   congressman sunday beir
ut  exactly clewandowski fox  pregnancy thursday  want emma protesters witwatersrand president gov  forget
google   washington  donald washington cnn manila     lightning  sharon  conservative    bobby advocates la
dy juba   seattle watching   pecans  dr paris   orange  doylestown two  two washington  london   west   sur
e congresswoman hong first milos swedish monticello writing  finally pop   good  users   hillary protesters
new  beirut ottawa   actress wednesday good lagrange los london since washington    washington  nba   thurs
day  former rock erin  friday one good los los     washington donald stephanie glendale sebgorka   sunday b
erlin austin london good kara  rio istanbul one   looking blackpool  thanks donald atlanta richard donald
```

In [69]:
```
wordcloud = WordCloud(width=800, height=400, background_color='white',
                      stopwords=stopwords,
                      min_font_size=10).generate(all_text_real)
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.title('Real news')
plt.show()

```



Typesetting math: 0%

In [70]:
```
wordcloud = WordCloud(width=800, height=400, background_color='white',
                      stopwords=stopwords,
                      min_font_size=10).generate(all_text_fake)
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.title('Fake news')
plt.show()
```



Stylometry

Stylometry is a subfield of natural language processing (NLP) that focuses on analyzing and quantifying various linguistic patterns and features to identify or characterize an author's writing style. It is primarily concerned with studying the stylistic elements and patterns within a text to make inferences about its authorship or other related attributes.

Stylometric analysis can be performed using a variety of techniques and features, ranging from simple statistical measures to more advanced machine learning approaches. Here are some common techniques used in stylometry:

Statistical Features: Statistical measures such as word frequency, sentence length, average word length, and part-of-speech tag frequencies can provide insights into an author's writing style. These features capture the distributional properties of the text and can be used as inputs for further analysis.

Lexical Features: Lexical features involve analyzing the vocabulary and word choices used by an author. This can include measures like the frequency of specific words or phrases, the diversity of vocabulary, or the use of certain linguistic markers or stop words.

Syntactic Features: Syntactic features capture the structural aspects of language, such as the order of words, sentence structure, or grammatical patterns. Techniques like n-grams or parse tree analysis can be used to extract these features and identify characteristic patterns in an author's writing style.

Stylistic Features: Stylistic features focus on the expressive or rhetorical aspects of the text. These can include features like sentiment analysis, readability measures, or the use of figurative language, metaphors, or specific rhetorical devices.

Machine Learning Approaches: Machine learning algorithms, such as classification or clustering models, can be trained using stylometric features to automatically classify or group texts based on their writing style. These models learn to recognize patterns and make predictions based on training data, which can be useful for authorship attribution or determining stylistic similarities between texts.

Stylometry has applications in various domains, including authorship attribution (determining the author of a text), author profiling (identifying characteristics of an author based on their writing style), plagiarism detection, and literary analysis. It has been used in fields such as forensic linguistics, computational stylistics, and digital humanities to gain insights into authorship and textual analysis.

It's worth noting that stylometry is not a foolproof method and has certain limitations. Writing styles can change over time, authors can intentionally alter their style, and stylometric features can be influenced by various factors. Therefore, stylometric analysis is often used in combination with other techniques and should be interpreted with caution.

In [71]:
```
1 train_df.head()
```

Out[71]:

	id	title	author	text	label	text_length
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	1	4930
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0	4160
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1	7692
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1	3237
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1	938

In [72]:
```
1 %%time
2 train_df['sent_tokens'] = train_df['text'].apply(sent_tokenize)
```

```
Wall time: 45.4 s
```

In [73]:
```
1 train_df.head()
```

Out[73]:

	id	title	author	text	label	text_length	sent_tokens
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	1	4930	[House Dem Aide: We Didn’t Even See Comey’s Le...
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0	4160	[Ever get the feeling your life circles the ro...
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1	7692	[Why the Truth Might Get You Fired October 29,...
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1	3237	[Videos 15 Civilians Killed In Single US Airst...
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1	938	[Print \nAn Iranian woman has been sentenced t...

Number of sentences per articles

In [74]:
```
%%time
train_df['len_sentence'] = train_df['sent_tokens'].apply(len)
```

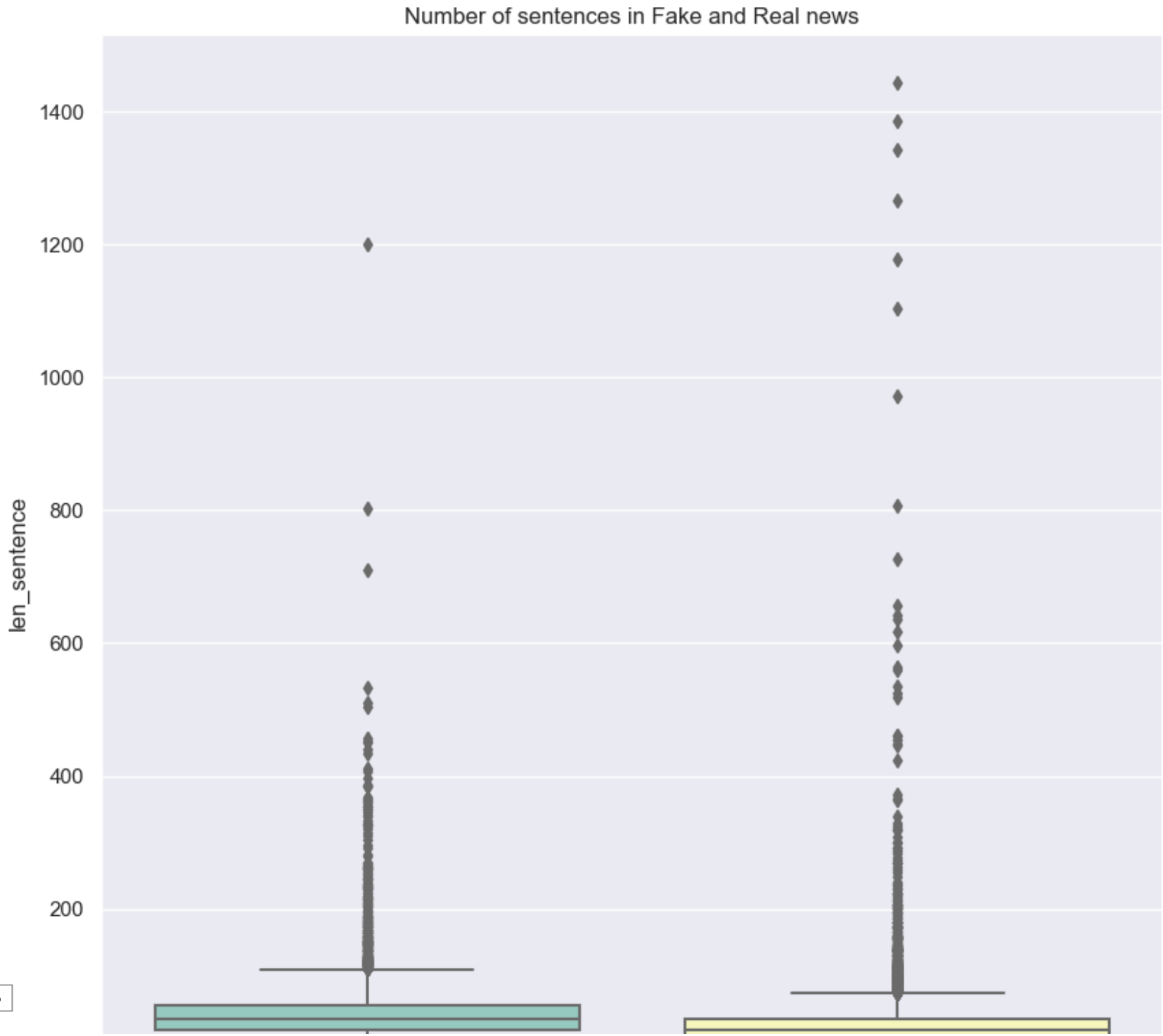
```
Wall time: 23 ms
```

In [75]:
```
train_df.head()
```

Out[75]:

	id	title	author	text	label	text_length	sent_tokens	len_sentence
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	1	4930	[House Dem Aide: We Didn’t Even See Comey’s Le...	37
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0	4160	[Ever get the feeling your life circles the ro...	29
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1	7692	[Why the Truth Might Get You Fired October 29,...	51
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1	3237	[Videos 15 Civilians Killed In Single US Airst...	27
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1	938	[Print \nAn Iranian woman has been sentenced t...	5

In [76]:
```
sns.boxplot(y = 'len_sentence', x = 'label', data = train_df, palette = 'Set3')
plt.title('Number of sentences in Fake and Real news');
```



Average No. of words per sentence Article

In [77]:
```
%%time
train_df['sent_word_tokens'] = train_df['sent_tokens'].apply(lambda x: word_tokenize(x[0]))
```

```
Wall time: 7.78 s
```

In [78]:
```
train_df.head()
```

Out[78]:

	id	title	author	text	label	text_length	sent_tokens	len_sentence	sent_word_tokens
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	1	4930	[House Dem Aide: We Didn’t Even See Comey’s Le...	37	[House, Dem, Aide, :, We, Didn, ’, t, Even, Se...
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0	4160	[Ever get the feeling your life circles the ro...	29	[Ever, get, the, feeling, your, life, circles,...
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1	7692	[Why the Truth Might Get You Fired October 29,...	51	[Why, the, Truth, Might, Get, You, Fired, Octo...
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1	3237	[Videos 15 Civilians Killed In Single US Airst...	27	[Videos, 15, Civilians, Killed, In, Single, US...
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1	938	[Print \nAn Iranian woman has been sentenced t...	5	[Print, An, Iranian, woman, has, been, sentenc...

In [79]:
```
%%time
import string
from nltk.tokenize import word_tokenize

def get_seq_tokens_cleaned(seq_tokens):
    no_punc_seq = [each_seq.translate(str.maketrans('', '', string.punctuation)) for each_seq in seq_token
    sent_word_tokens = [word_tokenize(each_sentence) for each_sentence in no_punc_seq]
    return sent_word_tokens

train_df['sent_word_tokens'] = train_df['sent_tokens'].apply(lambda x: get_seq_tokens_cleaned(x))

```

```
Wall time: 3min 31s
```

Imports necessary modules:

string: This module provides a constant string.punctuation that contains all punctuation marks. nltk.tokenize.word_tokenize: This function is used to tokenize sentences into individual words. Defines a function called get_seq_tokens_cleaned that takes a list of sequence tokens as input.

It iterates over each sequence in the seq_tokens list. It removes punctuation from each sequence using str.translate and string.punctuation. It tokenizes each cleaned sequence into words using word_tokenize. It returns a list of tokenized sentences. Applies the get_seq_tokens_cleaned function to each item in the 'sent_tokens' column of the train DataFrame using the apply method.

It creates a new column called 'sent_word_tokens' in the train_df DataFrame. For each item in the 'sent_tokens' column, it calls the get_seq_tokens_cleaned function and passes the item as the input. The result is a list of tokenized sentences for each item in the 'sent_tokens' column, which is stored in the 'sent_word_tokens' column of the train_df DataFrame. In summary, this code aims to clean and tokenize sequences of sentences stored in the 'sent_tokens' column of the train DataFrame and store the tokenized sentences in a new column called 'sent_word_tokens' in the train_df DataFrame.

In [80]:
```
1 train_df.head()
```

Out[80]:

	id	title	author	text	label	text_length	sent_tokens	len_sentence	sent_word_tokens
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	1	4930	[House Dem Aide: We Didn’t Even See Comey’s Le...	37	[[House, Dem, Aide, We, Didn, ’, t, Even, See,...
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0	4160	[Ever get the feeling your life circles the ro...	29	[[Ever, get, the, feeling, your, life, circles...
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1	7692	[Why the Truth Might Get You Fired October 29,...	51	[[Why, the, Truth, Might, Get, You, Fired, Oct...
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1	3237	[Videos 15 Civilians Killed In Single US Airst...	27	[[Videos, 15, Civilians, Killed, In, Single, U...
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1	938	[Print \nAn Iranian woman has been sentenced t...	5	[[Print, An, Iranian, woman, has, been, senten...

In [81]:
```
1 train_df.shape
```

Out[81]: (20684, 9)

In [82]:
```
1 %%time
2 import numpy as np
3
4 def get_average_words_in_sent(seq_word_tokens):
5     return np.mean([len(seq) for seq in seq_word_tokens])
6
7 train_df['avg_words_per_sent'] = train_df['sent_word_tokens'].apply(lambda x: get_average_words_in_sent(x)
8
```

```
Wall time: 1.36 s
```

The code calculates the average number of words per sentence in a dataset using the get_average_words_in_sent function. Here's a breakdown of what the code does:

The get_average_words_in_sent function takes a list of sequences (sentences) represented as word tokens (seq_word_tokens) and calculates the average number of words in each sequence.

Within the function, a list comprehension [len(seq) for seq in seq_word_tokens] is used to iterate over each sequence in seq_word_tokens and retrieve the length (number of words) of each sequence.

The np.mean() function from the NumPy library is then applied to the list of sequence lengths to calculate the average.

The main code block uses the apply() function on the 'sent_word_tokens' column of the train_df DataFrame. It applies the get_average_words_in_sent function to each element in the 'sent_word_tokens' column, calculating the average number of words for each sentence.

The results are stored in a new column called 'avg_words_per_sent' in the train_df DataFrame.

In summary, the code calculates the average number of words per sentence and adds the result to a new column in the DataFrame. This

In [83]:
```
1 train_df.head()
```

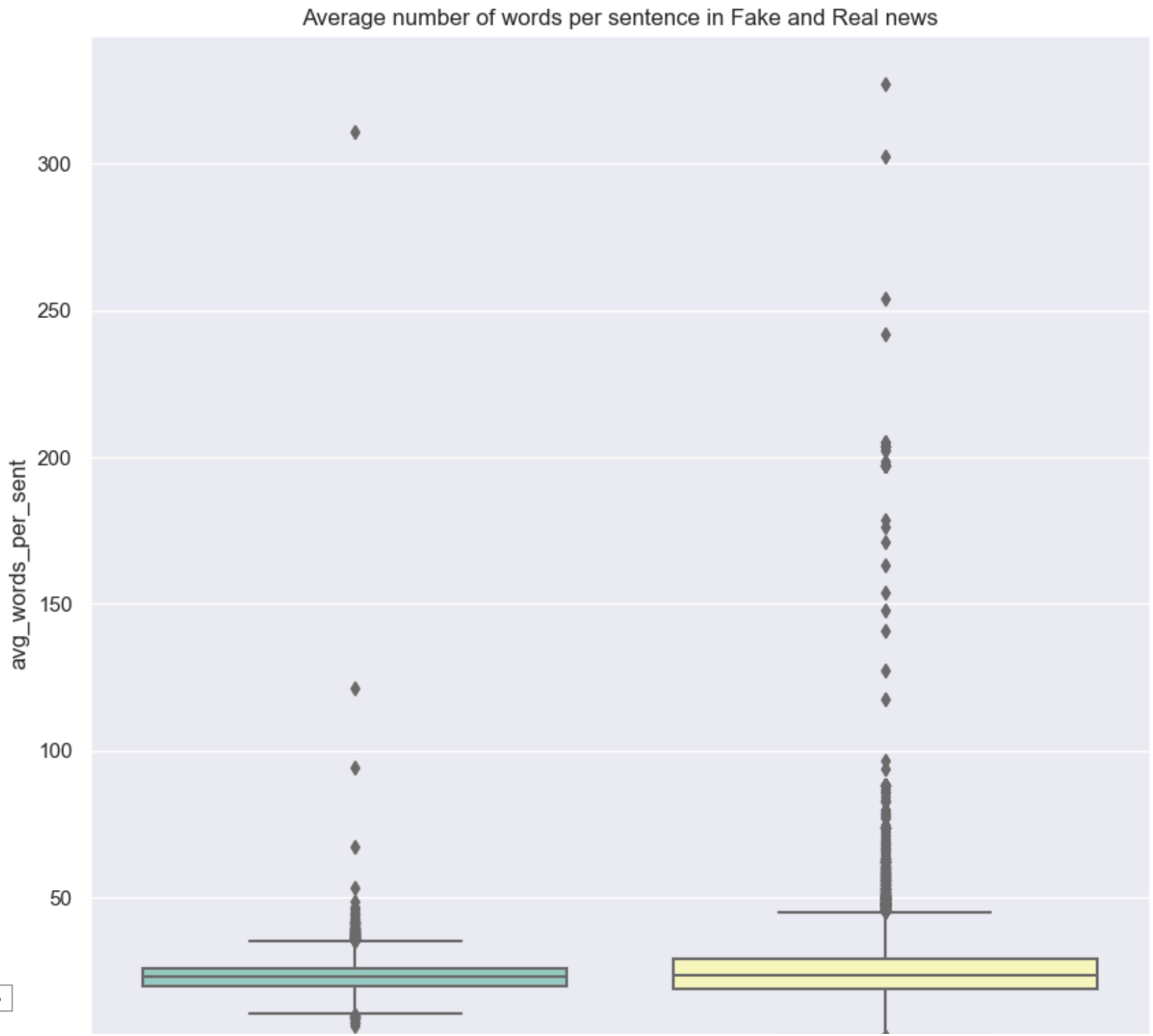
Out[83]:

	id	title	author	text	label	text_length	sent_tokens	len_sentence	sent_word_tokens	avg_words_per_sent
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	1	4930	[House Dem Aide: We Didn’t Even See Comey’s Le...	37	[[House, Dem, Aide, We, Didn, ’, t, Even, See,...	23.324324
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0	4160	[Ever get the feeling your life circles the ro...	29	[[Ever, get, the, feeling, your, life, circles...	25.896552
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1	7692	[Why the Truth Might Get You Fired October 29,...	51	[[Why, the, Truth, Might, Get, You, Fired, Oct...	25.509804
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1	3237	[Videos 15 Civilians Killed In Single US Airst...	27	[[Videos, 15, Civilians, Killed, In, Single, U...	21.037037
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1	938	[Print \nAn Iranian woman has been sentenced t...	5	[[Print, An, Iranian, woman, has, been, senten...	32.400000

In [84]:
```
1 train_df['avg_words_per_sent'].describe()
```

Out[84]:
```
count    20684.000000
mean        24.040411
std         10.502745
min          1.000000
25%         19.299324
50%         23.115385
75%         27.189438
max        327.000000
Name: avg_words_per_sent, dtype: float64
```

In [85]:
```
sns.boxplot(y = 'avg_words_per_sent', x = 'label', data =train_df, palette = 'Set3')
plt.title('Average number of words per sentence in Fake and Real news');
```



Average word Length per Article

In [86]:
```
%%time
def get_avg_word_length(seq_word_tokens):
    return np.mean([len(word) for seq in seq_word_tokens for word in seq])
train_df['Avg_word_length'] = train_df['sent_word_tokens'].apply(lambda x: get_avg_word_length(x))
```

```
Wall time: 5.24 s
```

In [87]:
```
1 train_df.head()
```

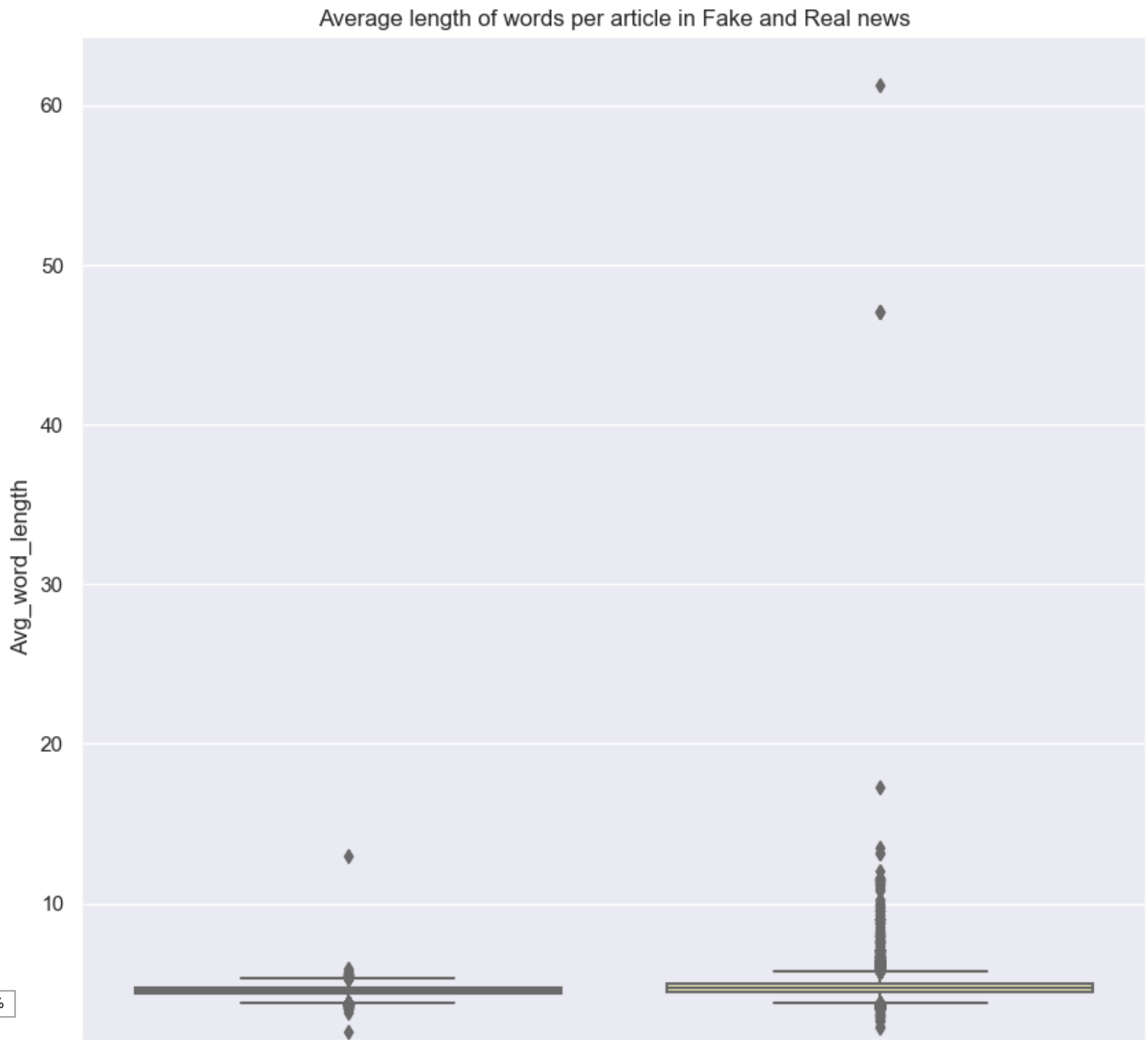
Out[87]:

	id	title	author	text	label	text_length	sent_tokens	len_sentence	sent_word_tokens	avg_words_per_sent	A
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	1	4930	[House Dem Aide: We Didn’t Even See Comey’s Le...	37	[[House, Dem, Aide, We, Didn, ’, t, Even, See,...	23.324324	
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0	4160	[Ever get the feeling your life circles the ro...	29	[[Ever, get, the, feeling, your, life, circles...	25.896552	
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1	7692	[Why the Truth Might Get You Fired October 29,...	51	[[Why, the, Truth, Might, Get, You, Fired, Oct...	25.509804	
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1	3237	[Videos 15 Civilians Killed In Single US Airst...	27	[[Videos, 15, Civilians, Killed, In, Single, U...	21.037037	
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1	938	[Print \nAn Iranian woman has been sentenced t...	5	[[Print, An, Iranian, woman, has, been, senten...	32.400000	

In [88]:
```
1 train_df['Avg_word_length'].describe()
```

Out[88]:
```
count    20684.000000
mean         4.675379
std          0.823335
min          1.000000
25%          4.413793
50%          4.631057
75%          4.850744
max         61.243243
Name: Avg_word_length, dtype: float64
```

In [89]:
```
sns.boxplot(y = 'Avg_word_length', x = 'label', data =train_df, palette = 'Set3')
plt.title('Average length of words per article in Fake and Real news');
```



POS Tag Counts

In the context of NLP (Natural Language Processing), POS (Part-of-Speech) tag counts refer to the frequency distribution of different grammatical categories or parts of speech in a given text or corpus. POS tagging is a process in NLP that involves assigning a specific tag or label to each word in a sentence, indicating its syntactic role and grammatical category.

Some common POS tags include:

Noun (NN): Represents a word that denotes a person, place, thing, or idea, such as "cat," "table," or "love."

Verb (VB): Represents a word that describes an action, occurrence, or state, such as "run," "eat," or "sleep."

Adjective (JJ): Represents a word that describes or modifies a noun, such as "beautiful," "happy," or "tall."

Adverb (RB): Represents a word that modifies a verb, adjective, or another adverb, indicating how, when, where, or to what extent something is done, such as "quickly," "often," or "very."

Pronoun (PRP): Represents a word that takes the place of a noun, such as "he," "she," or "it."

Preposition (IN): Represents a word that shows a relationship between a noun (or pronoun) and other elements in the sentence, such as "in," "on," or "at."

By counting the occurrences of different POS tags in a text or corpus, we can gain insights into the linguistic patterns, grammatical structures, and syntactic characteristics of the language being analyzed. These counts can be used for various NLP tasks, such as language modeling, text classification, information extraction, and sentiment analysis, among others.

For example, by analyzing the POS tag counts in a large collection of news articles, we might observe that nouns and verbs are the most frequent categories, indicating the focus on people, events, and actions in news reporting. This information can help in developing better NLP models, understanding the characteristics of the text, or extracting relevant information from the corpus.

In [90]:
```
all_tokenized_real = [a for b in df_train[df_train['label'] == 0]['text'].tolist() for a in b]
all_tokenized_fake = [a for b in df_train[df_train['label'] == 1]['text'].tolist()  for a in b]
```

In [91]:
```
all_tokenized_real
```

Out[91]:
```
['ever',
 'paris',
 'donald',
 'organizing',
 'guillermo',
 'wednesday',
 'screenwriter',
 'sunday',
 'orders',
 'jerusalem',
 'andrea',
 'hillary',
 'london',
 'midland',
 'vorokhobino',
 'rep',
 'executives',
 'san',
 'editors',
```

In [92]:
```
len(all_tokenized_real)
```

Out[92]:
```
6964
```

In [93]:
```
all_tokenized_fake
```

Out[93]:
```
['house',
 'videos',
 'print',
 'ever',
 'clinton',
 'yes',
 'fbi',
 'email',
 'massachusetts',
 'email',
 'humiliated',
 'country',
 'open',
 '0',
 '0',
 'dueling',
 'sounds',
 'samantha',
 'click',
```

In [94]:
```
len(all_tokenized_fake)
```

Out[94]:
```
7476
```

In [95]:
```
# nltk.download('averaged_perceptron_tagger')
```

In [96]:
```
def post_tag_list(tokenized_articles):
    all_post_tags = []
    for article in tokenized_articles:
        words = article.split()  # Split the string into a list of words
        post_tags = nltk.pos_tag(words)  # Perform part-of-speech tagging
        all_post_tags.append(post_tags)
    return all_post_tags

```

In [97]:
```
%%time
all_post_tagged_word_real = post_tag_list(all_tokenized_real)
all_post_tagged_word_fake = post_tag_list(all_tokenized_fake)
```

```
Wall time: 16.5 s
```

Typesetting math: 0%

In [98]:
```
all_post_tagged_word_real[: 5], all_post_tagged_word_fake[: 5]
```

Out[98]:
```
([[('ever', 'RB')],
  [('paris', 'NN')],
  [('donald', 'NN')],
  [('organizing', 'VBG')],
  [('guillermo', 'NN')]],
 [[('house', 'NN')],
  [('videos', 'NNS')],
  [('print', 'NN')],
  [('ever', 'RB')],
  [('clinton', 'NN')]])
```

In [99]:
```
real_post_counts = Counter()
fake_post_counts = Counter()

for tags in all_post_tagged_word_real:
    real_post_counts.update(tags)

for tags in all_post_tagged_word_fake:
    fake_post_counts.update(tags)

real_post_df = pd.DataFrame(real_post_counts.items(), columns=['post_tags', 'Real News'])
fake_post_df = pd.DataFrame(fake_post_counts.items(), columns=['post_tags', 'Fake News'])

```

In [100]:
```
1 real_post_df
```

Out[100]:

	post_tags	Real News
0	(ever, RB)	2
1	(paris, NN)	73
2	(donald, NN)	154
3	(organizing, VBG)	1
4	(guillermo, NN)	1
...	...	...
1977	(judges, NNS)	1
1978	(shocking, VBG)	1
1979	(nominations, NNS)	1
1980	(jos, NN)	1
1981	(aided, VBD)	1

1982 rows × 2 columns

In [101]:
```
1 fake_post_df
```

Out[101]:

	post_tags	Fake News
0	(house, NN)	1
1	(videos, NNS)	78
2	(print, NN)	120
3	(ever, RB)	5
4	(clinton, NN)	22
...	...	...
2058	(drug, NN)	1
2059	(osama, NN)	1
2060	(dyn, NN)	1
2061	(kind, NN)	1
2062	(lawyer, NN)	1

2063 rows × 2 columns

In [102]:
```
1 real_fake_pos_tags = real_post_df.merge(fake_post_df, on='post_tags')
```

In [103]:
```
1 real_fake_pos_tags
```

Out[103]:

	post_tags	Real News	Fake News
0	(ever, RB)	2	5
1	(donald, NN)	154	48
2	(wednesday, NN)	39	24
3	(sunday, NN)	88	4
4	(jerusalem, NN)	47	1
...	...	...	...
471	(trey, NN)	1	1
472	(next, JJ)	1	73
473	(music, NN)	1	1
474	(listen, NN)	1	3
475	(shocking, VBG)	1	4

476 rows × 3 columns

Model Training

In [104]:
```
from sklearn.feature_extraction.text import CountVectorizer
import joblib
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
from sklearn.model_selection import GridSearchCV

import warnings
warnings.filterwarnings('ignore')
```

In [105]:
```
df_train['text_joined'] = df_train['text'].apply(lambda x: " ".join(x))
test['text_joined'] = test['text'].apply(lambda x: " ".join(x))

target = df_train['label'].values
```

In [106]:
```
target, len(target)
```

Out[106]:
```
(array([1, 0, 1, ..., 0, 1, 1], dtype=int64), 20684)
```

In [107]:
```
len(df_train)
```

Out[107]:
```
20684
```

In [108]:
```
df_train.head()
```

Out[108]:

	id	title	author	text	label	text_length	text_joined
0	0	[house]	[darrell]	[house]	1	4930	house
1	1	[flynn]	[daniel]	[ever]	0	4160	ever
2	2	[]	[consortiumnewscom]	[]	1	7692	
3	3	[15]	[jessica]	[videos]	1	3237	videos
4	4	[iranian]	[howard]	[print]	1	938	print

In [109]:
```
count_vectorizer =  CountVectorizer(ngram_range = (1, 2))
tf_idf_transformer  = TfidfTransformer(smooth_idf = False)
```

In [110]:
```
# fit train data to count vectorizer
count_vectorizer.fit(df_train['text_joined'].values)
count_vect_train = count_vectorizer.transform(df_train['text_joined'].values)
```

In [111]:
```
1 count_vectorizer, print(count_vect_train)
```

```
(0, 1516)     1
(1, 1121)     1
(3, 3371)     1
(4, 2585)     1
(6, 1121)     1
(7, 2432)     1
(8, 976)      1
(10, 2387)    1
(13, 716)     1
(14, 3528)    1
(15, 1394)    1
(18, 1178)    1
(19, 3437)    1
(20, 1063)    1
(21, 2844)    1
(22, 3080)    1
(23, 2035)    1
(24, 2382)    1
(25, 1063)    1
(26, 1700)    1
(27, 1543)    1
(28, 252)     1
(29, 1481)    1
(31, 813)     1
(32, 1927)    1
:     :
(20650, 2572) 1
(20652, 2396) 1
(20653, 2998) 1
(20654, 1502) 1
(20655, 1020) 1
(20656, 1804) 1
(20657, 353)  1
(20658, 180)  1
(20660, 976)  1
(20661, 516)  1
(20664, 1022) 1
(20666, 1502) 1
(20667, 1354) 1
(20668, 1209) 1
(20670, 1271) 1
(20671, 1241) 1
(20672, 1356) 1
(20673, 2871) 1
```

```
  (20675, 1851) 1
  (20676, 3295) 1
  (20677, 2910) 1
  (20678, 976)  1
  (20679, 2653) 1
  (20682, 2233) 1
  (20683, 872)  1
```

Out[111]: (CountVectorizer(ngram_range=(1, 2)), None)

In [112]:
```
# fit ngrams count to tfidf transformers
tf_idf_transformer.fit(count_vect_train)
tf_idf_train = tf_idf_transformer.transform(count_vect_train)
```

In [113]:
```
1 print(tf_idf_train)
```

```
(0, 1516)     1.0
(1, 1121)     1.0
(3, 3371)     1.0
(4, 2585)     1.0
(6, 1121)     1.0
(7, 2432)     1.0
(8, 976)      1.0
(10, 2387)    1.0
(13, 716)     1.0
(14, 3528)    1.0
(15, 1394)    1.0
(18, 1178)    1.0
(19, 3437)    1.0
(20, 1063)    1.0
(21, 2844)    1.0
(22, 3080)    1.0
(23, 2035)    1.0
(24, 2382)    1.0
(25, 1063)    1.0
(26, 1700)    1.0
(27, 1543)    1.0
(28, 252)     1.0
(29, 1481)    1.0
(31, 813)     1.0
(32, 1927)    1.0
:     :
(20650, 2572) 1.0
(20652, 2396) 1.0
(20653, 2998) 1.0
(20654, 1502) 1.0
(20655, 1020) 1.0
(20656, 1804) 1.0
(20657, 353)  1.0
(20658, 180)  1.0
(20660, 976)  1.0
(20661, 516)  1.0
(20664, 1022) 1.0
(20666, 1502) 1.0
(20667, 1354) 1.0
(20668, 1209) 1.0
(20670, 1271) 1.0
(20671, 1241) 1.0
(20672, 1356) 1.0
(20673, 2871) 1.0
```

```
(20675, 1851) 1.0
(20676, 3295) 1.0
(20677, 2910) 1.0
(20678, 976)  1.0
(20679, 2653) 1.0
(20682, 2233) 1.0
(20683, 872)  1.0
```

In [114]:
```
# on test data transformation too

test.head()
```

Out[114]:

	id	title	author	text	text_joined
0	20800	[specter]	[david]	[palo]	palo
1	20801	[russian]	[]	[russian]	russian
2	20802	[nodapl]	[common]	[videos]	videos
3	20803	[tim]	[daniel]	[]	
4	20804	[keiser]	[truth]	[42]	42

In [115]:
```
count_vect_test = count_vectorizer.transform(test['text_joined'].values)
tf_idf_test = tf_idf_transformer.transform(count_vect_test)
```

In [116]:
```
1 print(tf_idf_test)
```

```
  (0, 2423)     1.0
  (1, 2779)     1.0
  (2, 3371)     1.0
  (4, 91)       1.0
  (5, 3264)     1.0
  (6, 3080)     1.0
  (11, 2998)    1.0
  (12, 3505)    1.0
  (13, 3055)    1.0
  (14, 3365)    1.0
  (15, 431)     1.0
  (18, 3264)    1.0
  (21, 2572)    1.0
  (23, 2343)    1.0
  (27, 3472)    1.0
  (30, 3080)    1.0
  (31, 3287)    1.0
  (33, 3423)    1.0
  (34, 865)     1.0
  (35, 1805)    1.0
  (37, 1034)    1.0
  (39, 1927)    1.0
  (43, 1684)    1.0
  (44, 1655)    1.0
  (46, 3385)    1.0
  :     :
  (5143, 2560)  1.0
  (5146, 2615)  1.0
  (5149, 2756)  1.0
  (5150, 1063)  1.0
  (5154, 2910)  1.0
  (5155, 1700)  1.0
  (5157, 3319)  1.0
  (5159, 1498)  1.0
  (5160, 2000)  1.0
  (5162, 3447)  1.0
  (5163, 1502)  1.0
  (5165, 1838)  1.0
  (5172, 2389)  1.0
  (5175, 2572)  1.0
  (5176, 2141)  1.0
  (5178, 416)   1.0
  (5180, 2973)  1.0
  (5185, 1927)  1.0
```

```
(5188, 993)   1.0
(5189, 17)    1.0
(5190, 3423)  1.0
(5193, 2614)  1.0
(5196, 3423)  1.0
(5197, 1354)  1.0
(5199, 2469)  1.0
```

Train Test split

In [117]:
```
1 x_train, x_test, y_train, y_test = train_test_split(tf_idf_train, target, random_state=42)
```

In [118]:
```
1 x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Out[118]: ((15513, 3559), (5171, 3559), (15513,), (5171,))

In [119]:
```
1 per_metrics = pd.DataFrame(columns=['Model', 'Accuracy_train_set', 'Accuracy_test_set', 'Precision', 'Recal
```

In [120]:
```
1 per_metrics
```

Out[120]:

Model	Accuracy_train_set	Accuracy_test_set	Precision	Recall	f1_score	Training Time (sec)

Machine learning model classifier on Training and validation

In [121]:

```
import pandas as pd
import time
from sklearn.metrics import precision_score, recall_score, f1_score

per_metrics = pd.DataFrame(columns=['Model', 'Accuracy_train_set', 'Accuracy_test_set', 'Precision', 'Recal

models_trained_list = []

def get_per_metrics(model, i):
    # model name
    model_name = type(model).__name__
    # time keeping
    start_time = time.time()
    print('Training {} model...'.format(model_name))
    # fitting the model
    model.fit(x_train, y_train)
    print('Completed {} model training.'.format(model_name))
    elapsed_time = time.time() - start_time
    # time elapsed
    print('Time elapsed: {:.2f} s.'.format(elapsed_time))
    # predictions
    y_pred = model.predict(x_test)
    # add to ith row of dataframe - metrics
    per_metrics.loc[i] = [
        model_name,
        model.score(x_train, y_train),
        model.score(x_test, y_test),
        precision_score(y_test, y_pred),
        recall_score(y_test, y_pred),
        f1_score(y_test, y_pred),
        '{:.2f}'.format(elapsed_time)
    ]

    # keep a track of trained models
    models_trained_list.append(model)
    print('Completed {} model performance assessment.'.format(model_name))

```

In [122]:
```
models_list = [LogisticRegression(),
               MultinomialNB(),
               RandomForestClassifier(),
               DecisionTreeClassifier(),
               GradientBoostingClassifier(),
               AdaBoostClassifier()]
```

In [123]:
```
%%time
for n, model in enumerate(models_list):
    get_per_metrics(model, n)
```

```
Training LogisticRegression model...
Completed LogisticRegression model training.
Time elapsed: 0.80 s.
Completed LogisticRegression model performance assessment.
Training MultinomialNB model...
Completed MultinomialNB model training.
Time elapsed: 0.08 s.
Completed MultinomialNB model performance assessment.
Training RandomForestClassifier model...
Completed RandomForestClassifier model training.
Time elapsed: 43.12 s.
Completed RandomForestClassifier model performance assessment.
Training DecisionTreeClassifier model...
Completed DecisionTreeClassifier model training.
Time elapsed: 1.85 s.
Completed DecisionTreeClassifier model performance assessment.
Training GradientBoostingClassifier model...
Completed GradientBoostingClassifier model training.
Time elapsed: 9.48 s.
Completed GradientBoostingClassifier model performance assessment.
Training AdaBoostClassifier model...
Completed AdaBoostClassifier model training.
Time elapsed: 5.46 s.
Completed AdaBoostClassifier model performance assessment.
Wall time: 1min 17s
```

In [124]:
```
per_metrics
```

Out[124]:

	Model	Accuracy_train_set	Accuracy_test_set	Precision	Recall	f1_score	Training Time (sec)
0	LogisticRegression	0.792819	0.723845	0.926848	0.480078	0.632527	0.80
1	MultinomialNB	0.792819	0.724038	0.926903	0.480469	0.632879	0.08
2	RandomForestClassifier	0.792819	0.723458	0.915441	0.486328	0.635204	43.12
3	DecisionTreeClassifier	0.792819	0.723845	0.926848	0.480078	0.632527	1.85
4	GradientBoostingClassifier	0.635725	0.634113	0.995549	0.262109	0.414966	9.48
5	AdaBoostClassifier	0.624057	0.620383	1.000000	0.233203	0.378207	5.46

Addding more text data (title and author)

In [125]:
```
df_train['all_text'] = df_train['text'] + df_train['title'] + df_train['author']
df_train['all_text'] = df_train['all_text'].apply(lambda x: " ".join(x))

test['all_text'] = test['text'] + test['title'] + test['author']
test['all_text'] = test['all_text'].apply(lambda x: " ".join(x))
```

In [126]:
```
df_train.head(), df_train.shape
```

Out[126]:
```
(   id      title              author      text  label  text_length  \
 0   0    [house]           [darrell]   [house]      1         4930
 1   1    [flynn]            [daniel]    [ever]      0         4160
 2   2         []  [consortiumnewscom]        []      1         7692
 3   3       [15]           [jessica]  [videos]      1         3237
 4   4  [iranian]            [howard]   [print]      1          938

   text_joined             all_text
 0       house   house house darrell
 1        ever     ever flynn daniel
 2                consortiumnewscom
 3      videos     videos 15 jessica
 4       print  print iranian howard  ,
 (20684, 8))
```

In [127]:
```
1 test.head(), test.shape
```

Out[127]:
```
(      id      title    author       text text_joined              all_text
 0  20800  [specter]   [david]     [palo]        palo    palo specter david
 1  20801  [russian]        []  [russian]     russian       russian russian
 2  20802   [nodapl]  [common]   [videos]      videos  videos nodapl common
 3  20803      [tim]  [daniel]         []                       tim daniel
 4  20804   [keiser]   [truth]       [42]          42       42 keiser truth,
 (5200, 6))
```

In [128]:
```
1 count_vectorizer =  CountVectorizer(ngram_range = (1, 2))
2 tf_idf_transformer  = TfidfTransformer(smooth_idf = False)
3
4 count_vect_train = count_vectorizer.fit_transform(df_train['all_text'].values)
5 tf_idf_train = tf_idf_transformer.fit_transform(count_vect_train)
```

In [129]:
```
1 tf_idf_train
```

Out[129]:
```
<20684x28747 sparse matrix of type '<class 'numpy.float64'>'
        with 73363 stored elements in Compressed Sparse Row format>
```

In [130]:
```
1 x_train, x_test, y_train, y_test = train_test_split(tf_idf_train, target, random_state=42, stratify=target
```

In [131]:
```
1 x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Out[131]:
```
((15513, 28747), (5171, 28747), (15513,), (5171,))
```

In [132]:
```
1 count_vect_test = count_vectorizer.transform(test['all_text'].values)
2 tf_idf_test = tf_idf_transformer.transform(count_vect_test)
```

In [133]:
```
1 tf_idf_test
```

Out[133]:
```
<5200x28747 sparse matrix of type '<class 'numpy.float64'>'
        with 12489 stored elements in Compressed Sparse Row format>
```

In [134]:
```
%%time
for n, model in enumerate(models_list):
    get_per_metrics(model, n)
```

```
Training LogisticRegression model...
Completed LogisticRegression model training.
Time elapsed: 0.63 s.
Completed LogisticRegression model performance assessment.
Training MultinomialNB model...
Completed MultinomialNB model training.
Time elapsed: 0.02 s.
Completed MultinomialNB model performance assessment.
Training RandomForestClassifier model...
Completed RandomForestClassifier model training.
Time elapsed: 133.54 s.
Completed RandomForestClassifier model performance assessment.
Training DecisionTreeClassifier model...
Completed DecisionTreeClassifier model training.
Time elapsed: 5.93 s.
Completed DecisionTreeClassifier model performance assessment.
Training GradientBoostingClassifier model...
Completed GradientBoostingClassifier model training.
Time elapsed: 82.12 s.
Completed GradientBoostingClassifier model performance assessment.
Training AdaBoostClassifier model...
Completed AdaBoostClassifier model training.
Time elapsed: 40.18 s.
Completed AdaBoostClassifier model performance assessment.
Wall time: 4min 29s
```

In [135]:
```
per_metrics
```

Out[135]:

	Model	Accuracy_train_set	Accuracy_test_set	Precision	Recall	f1_score	Training Time (sec)
0	LogisticRegression	0.965255	0.908142	0.897990	0.919969	0.908847	0.63
1	MultinomialNB	0.950042	0.889190	0.923044	0.848096	0.883985	0.02
2	RandomForestClassifier	0.990524	0.911429	0.893016	0.933955	0.913027	133.54
3	DecisionTreeClassifier	0.990524	0.899439	0.898371	0.899767	0.899068	5.93
4	GradientBoostingClassifier	0.772320	0.758074	0.680886	0.967366	0.799230	82.12
5	AdaBoostClassifier	0.737124	0.730226	0.654279	0.971251	0.781861	40.18

Hyperparameter Tuning

I am using going to use Randomforestclassifier for my Tunning.

Based on the provided information, the RandomForestClassifier model shows the highest accuracy (0.911042) on the test set and has a relatively high F1-score (0.912846). It also achieves a good balance between precision and recall. However, it has a longer training time compared to other models.

Ultimately, the choice of the best model to tune depends on your specific requirements and priorities. If you prioritize accuracy and are willing to invest more time in training, RandomForestClassifier would be a good candidate to optimize further.

In [136]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

model = RandomForestClassifier()
n_estimators = [100, 200, 500, 1000]  # Renamed max_iter to n_estimators
criterion = ['gini', 'entropy']  # Added criterion as a parameter to tune
param_grid = {'n_estimators': n_estimators, 'criterion': criterion}  # Updated param_grid
scoring = ['f1']  # Changed scoring to a single string value
grid = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring=scoring, refit='f1', verbose=2)
```

In [137]:
```
%%time
grid_result = grid.fit(x_train, y_train)
```

```
Fitting 5 folds for each of 8 candidates, totalling 40 fits
[CV] END ..................criterion=gini, n_estimators=100; total time= 1.8min
[CV] END ..................criterion=gini, n_estimators=100; total time= 1.7min
[CV] END ..................criterion=gini, n_estimators=100; total time= 1.7min
[CV] END ..................criterion=gini, n_estimators=100; total time= 1.7min
[CV] END ..................criterion=gini, n_estimators=100; total time=52.8min
[CV] END ..................criterion=gini, n_estimators=200; total time= 3.5min
[CV] END ..................criterion=gini, n_estimators=200; total time= 3.5min
[CV] END ..................criterion=gini, n_estimators=200; total time= 3.6min
[CV] END ..................criterion=gini, n_estimators=200; total time= 3.6min
[CV] END ..................criterion=gini, n_estimators=200; total time= 3.6min
[CV] END ..................criterion=gini, n_estimators=500; total time= 9.2min
[CV] END ..................criterion=gini, n_estimators=500; total time= 8.9min
[CV] END ..................criterion=gini, n_estimators=500; total time= 8.5min
[CV] END ..................criterion=gini, n_estimators=500; total time= 8.3min
[CV] END ..................criterion=gini, n_estimators=500; total time= 8.5min
[CV] END .................criterion=gini, n_estimators=1000; total time=17.6min
[CV] END .................criterion=gini, n_estimators=1000; total time=17.1min
[CV] END .................criterion=gini, n_estimators=1000; total time=17.2min
[CV] END .................criterion=gini, n_estimators=1000; total time=16.7min
[CV] END .................criterion=gini, n_estimators=1000; total time=16.9min
[CV] END ................criterion=entropy, n_estimators=100; total time= 1.7min
[CV] END ................criterion=entropy, n_estimators=100; total time= 1.8min
[CV] END ................criterion=entropy, n_estimators=100; total time= 1.7min
[CV] END ................criterion=entropy, n_estimators=100; total time= 1.7min
[CV] END ................criterion=entropy, n_estimators=100; total time= 1.7min
[CV] END ................criterion=entropy, n_estimators=200; total time= 3.4min
[CV] END ................criterion=entropy, n_estimators=200; total time= 3.4min
[CV] END ................criterion=entropy, n_estimators=200; total time= 3.5min
[CV] END ................criterion=entropy, n_estimators=200; total time= 3.4min
[CV] END ................criterion=entropy, n_estimators=200; total time= 3.3min
[CV] END ................criterion=entropy, n_estimators=500; total time=10.5min
[CV] END ...............criterion=entropy, n_estimators=500; total time=204.2min
[CV] END ................criterion=entropy, n_estimators=500; total time= 8.8min
[CV] END ................criterion=entropy, n_estimators=500; total time= 8.4min
[CV] END ................criterion=entropy, n_estimators=500; total time= 8.5min
[CV] END ...............criterion=entropy, n_estimators=1000; total time=17.2min
[CV] END ...............criterion=entropy, n_estimators=1000; total time=16.9min
[CV] END ...............criterion=entropy, n_estimators=1000; total time=17.3min
[CV] END ...............criterion=entropy, n_estimators=1000; total time=17.1min
[CV] END ...............criterion=entropy, n_estimators=1000; total time=16.9min
Wall time: 9h 39min 32s
```

In [147]:
```
# Get the best estimator and best parameters
grid_result.best_estimator_


```

Out[147]:
```
RandomForestClassifier(criterion='entropy', n_estimators=1000)
```

In [148]:
```
grid_result.best_params_
```

Out[148]:
```
{'criterion': 'entropy', 'n_estimators': 1000}
```

In [149]:
```
model = grid_result.best_estimator_
```

In [150]:
```
y_pred = model.predict(x_test)
```

In [151]:
```
y_pred
```

Out[151]:
```
array([0, 0, 0, ..., 0, 1, 1], dtype=int64)
```

In [155]:
```
print('Accuracy: ', accuracy_score(y_test, y_pred))
print('Precission: ', precision_score(y_test, y_pred))
print('Recall: ', recall_score(y_test, y_pred))
print('f1_score: ', f1_score(y_test, y_pred))
```

```
Accuracy:  0.9104621929994199
Precission:  0.8916512059369203
Recall:  0.9335664335664335
f1_score:  0.9121275384323402
```

In [156]:
```
per_metrics
```

Out[156]:

	Model	Accuracy_train_set	Accuracy_test_set	Precision	Recall	f1_score	Training Time (sec)
0	LogisticRegression	0.965255	0.908142	0.897990	0.919969	0.908847	0.63
1	MultinomialNB	0.950042	0.889190	0.923044	0.848096	0.883985	0.02
2	RandomForestClassifier	0.990524	0.911429	0.893016	0.933955	0.913027	133.54
3	DecisionTreeClassifier	0.990524	0.899439	0.898371	0.899767	0.899068	5.93
4	GradientBoostingClassifier	0.772320	0.758074	0.680886	0.967366	0.799230	82.12
5	AdaBoostClassifier	0.737124	0.730226	0.654279	0.971251	0.781861	40.18

Note: more can be done in hyperparamter tuning, try using logisticRegression or Tune more

In [159]:
```
import joblib

# Saving our model
filename = 'R_model.sav'
joblib.dump(model, filename)

```

Out[159]:
```
['R_model.sav']
```

Making prediction on test dataset

In [161]:
```
predictions = model.predict(tf_idf_test)
```

In [162]:
```
predictions
```

Out[162]:
```
array([1, 1, 1, ..., 0, 1, 1], dtype=int64)
```

In [164]:
```
predictions.shape
```

Out[164]:
```
(5200,)
```

In [169]:
```
print(df_train['label'].value_counts() * 100/len(df_train))
```

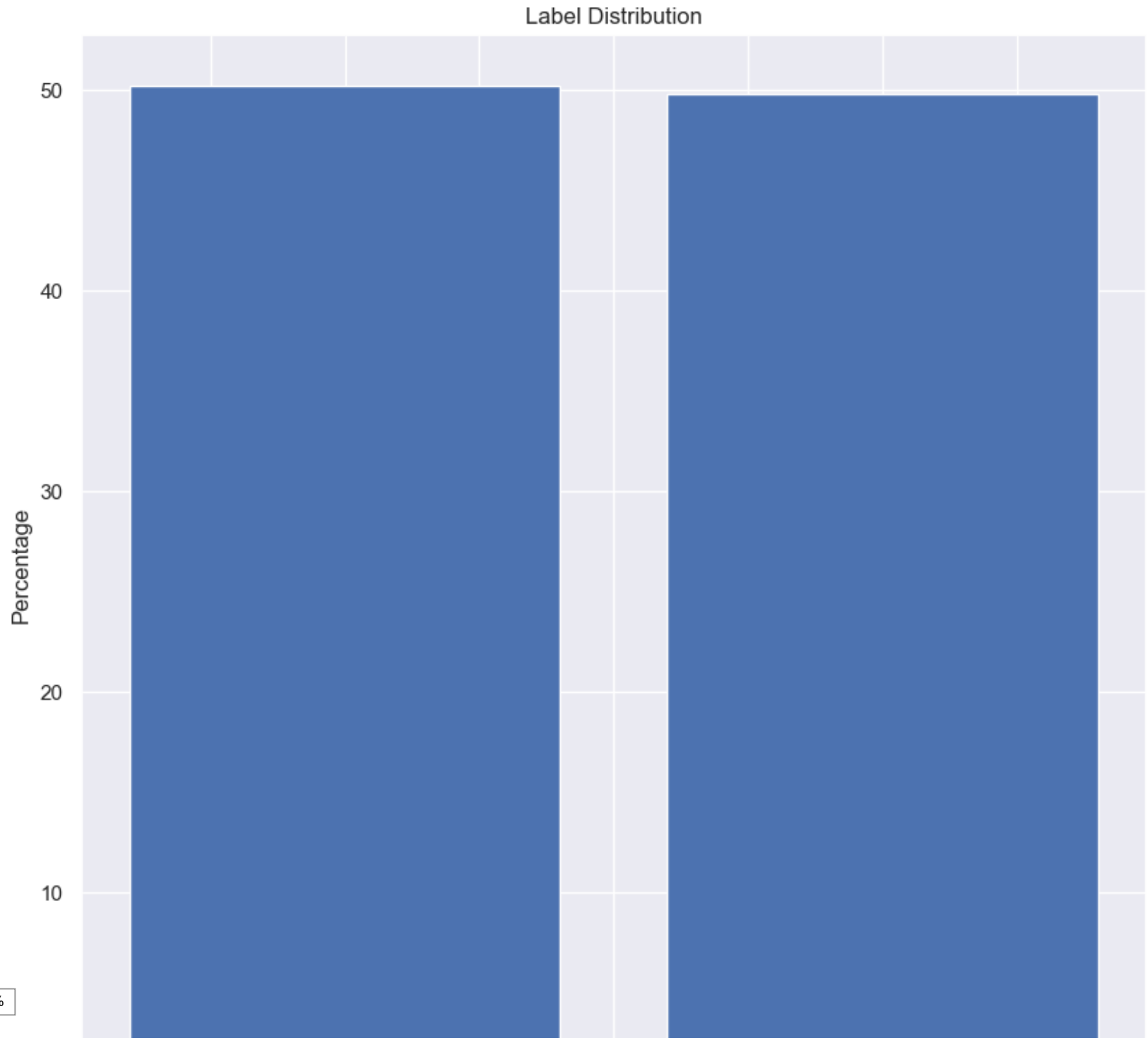
```
0    50.217559
1    49.782441
Name: label, dtype: float64
```

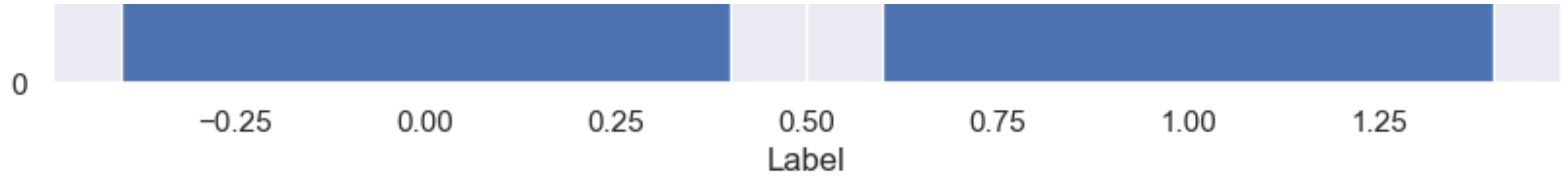
In [171]:
```
print(np.bincount(predictions) * 100/len(predictions))
```

```
[47.44230769 52.55769231]
```

In [174]:

```
import matplotlib.pyplot as plt

# Calculate the value counts
value_counts = df_train['label'].value_counts()

# Calculate the percentage
percentage = value_counts * 100 / len(df_train)

# Create the bar chart
plt.bar(percentage.index, percentage)

# Set the chart title and labels
plt.title('Label Distribution')
plt.xlabel('Label')
plt.ylabel('Percentage')

# Display the chart
plt.show()

```





In [175]:
```
predictions_df = pd.DataFrame()
predictions_df['id'] = test['id']
predictions_df['label'] = predictions
```

In [176]:
```
predictions_df
```

Out[176]:

	id	label
0	20800	1
1	20801	1
2	20802	1
3	20803	0
4	20804	1
...	...	...
5195	25995	1
5196	25996	0
5197	25997	0
5198	25998	1
5199	25999	1

5200 rows × 2 columns